

TIBCO ActiveSpaces®

C Reference

*Release 2.1.2 Add-On
January 2014*

Important Information

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE "LICENSE" FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document contains confidential information that is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIB, TIBCO, TIBCO Adapter, Predictive Business, Information Bus, The Power of Now, TIBCO ActiveMatrix BusinessEvents, and TIBCO ActiveSpaces are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

THIS SOFTWARE MAY BE AVAILABLE ON MULTIPLE OPERATING SYSTEMS. HOWEVER, NOT ALL OPERATING SYSTEM PLATFORMS FOR A SPECIFIC SOFTWARE VERSION ARE RELEASED AT THE SAME TIME. SEE THE README FILE FOR THE AVAILABILITY OF THIS SOFTWARE VERSION ON A SPECIFIC OPERATING SYSTEM PLATFORM.

THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

Copyright © 1999-2014 TIBCO Software Inc. ALL RIGHTS RESERVED.

TIBCO Software Inc. Confidential Information

Contents

Preface	xv
Related Documentation	xvi
TIBCO ActiveSpaces Documentation	xvi
Other TIBCO Product Documentation	xvi
Typographical Conventions	xvii
How to Contact TIBCO Support	xix
 Chapter 1 Introduction	 1
Overview	2
C API Header Files	3
Status Codes	4
Building Your Application	5
Running Your Application	7
 Chapter 2 Metaspace	 9
Metaspace Operations	10
tibasMetaspace_Connect()	14
tibasMetaspace_ConnectEx()	15
tibas_GetMetaspace()	16
tibas_GetMetaspaceNames()	17
tibasMetaspace_GetSpace()	18
tibasMetaspace_GetSpaceEx()	20
tibasMetaspace_GetSystemSpace()	21
tibasMetaspace_GetUserSpaceNames()	22
tibasMetaspace_DefineSpace()	23
tibasMetaspace_DropSpace()	25
tibasMetaspace_AlterSpace()	26
tibasMetaspace_RecoverSpace()	27
tibasMetaspace_RecoverSpaceEx()	28
tibasMetaspace_GetSpaceDef()	29
tibasMetaspace_Listen()	30
tibasMetaspace_ListenSpaceMemberEvents()	32
tibasMetaspace_ListenMemberEvents()	33
tibasMetaspace_ListenSpaceRemoteMemberEvents()	34
tibasMetaspace_ListenRemoteMemberEvents()	35
tibasMetaspace_ListenSpaceState()	36

tibasMetaspace_ListenSpaceDef()	37
tibasMetaspace_GetRemoteListen()	38
tibasMetaspace_Browse()	39
tibasMetaspace_BrowseEvents()	41
tibasMetaspace_BeginTransaction()	43
tibasMetaspace_CommitTransaction()	44
tibasMetaspace_RollbackTransaction()	45
tibasMetaspace_ReleaseContext()	46
tibasMetaspace_AcquireContext()	47
tibasMetaspace_GetRemoteMembers()	48
tibasMetaspace_GetMembers()	49
tibasMetaspace_GetSpaceMembers()	50
tibasMetaspace_GetSelfMember()	51
tibasMetaspace_GetLogLevel()	52
tibasMetaspace_GetDiscovery()	53
tibasMetaspace_GetListen()	54
tibasMetaspace_GetRemoteDiscovery()	55
tibasMetaspace_SetLogLevel()	56
tibasMetaspace_Free()	57
tibas_FreeData()	58
Chapter 3 Space	59
Space Operations and Options Initialization Operations	60
tibasSpace_Get()	66
tibasSpace_IsReady()	67
tibasSpace_WaitForReady()	68
tibasSpace_GetAll()	69
tibasSpace_Browse()	70
tibasSpace_BrowseEvents()	72
tibasSpace_CompareAndPut()	73
tibasSpace_CompareAndPutEx()	74
tibasSpace_CompareAndPutAll()	76
tibasSpace_CompareAndPutAllEx()	77
tibasSpace_CompareAndTake()	79
tibasSpace_CompareAndTakeEx()	80
tibasSpace_CompareAndTakeAll()	82
tibasSpace_CompareAndTakeAllEx()	83
tibasSpace_Listen()	85
tibasSpace_Put()	86
tibasSpace_PutEx()	87
tibasSpace_PutAll()	89
tibasSpace_PutAllEx()	90
tibasSpace_Take()	92
tibasSpace_TakeEx()	93

tibasSpace_TakeAll()	95
tibasSpace_TakeAllEx()	96
tibasSpace_GetSpaceDef()	98
tibasSpace_GetName()	99
tibasSpace_GetMetaspaceName()	100
tibasSpace_GetMetaspace()	101
tibasSpace_Size()	102
tibasSpace_Lock()	103
tibasSpace_LockEx()	104
tibasSpace_LockAll()	105
tibasSpace_LockAllEx()	106
tibasSpace_Unlock()	107
tibasSpace_UnlockAll()	108
tibasSpace_SetDistributionRole()	109
tibasSpace_SetPersister()	110
tibasSpace_Invoke()	111
tibasSpace_InvokeMember()	112
tibasSpace_InvokeMembers()	113
tibasSpace_InvokeRemoteMembers()	114
tibasSpace_InvokeSeeders()	115
tibasSpace_Load()	116
tibasSpace_LoadAll()	117
tibasSpace_Free()	118
tibasRecoveryOptions_Initialize()	119
tibasPutOptions_Initialize()	120
tibasTakeOptions_Initialize()	121
tibasLockOptions_Initialize()	122
tibasUnlockOptions_Initialize()	123
tibasInvokeOptions_Initialize()	124
Chapter 4 Admin	125
Admin Operations	126
tibasAdmin_Create()	127
tibasAdmin_Connect()	128
tibasAdmin_Execute()	129
tibasAdmin_Free()	131
Chapter 5 SpaceDef	133
SpaceDef Operations	134
tibasSpaceDef_Create()	138
tibasSpaceDef_GetName()	140
tibasSpaceDef_SetName()	141
tibasSpaceDef_GetReplicationCount()	142
tibasSpaceDef_SetReplicationCount()	143

tibasSpaceDef_GetCapacity()	144
tibasSpaceDef_SetCapacity()	145
tibasSpaceDef_SetEvictionPolicy()	146
tibasSpaceDef_GetEvictionPolicy()	147
tibasSpaceDef_GetLockScope()	148
tibasSpaceDef_SetLockScope()	149
tibasSpaceDef_GetNumFields()	150
tibasSpaceDef_GetIndex()	151
tibasSpaceDef_RemoveIndexDef()	152
tibasSpaceDef_GetMinSeederCount()	153
tibasSpaceDef_SetMinSeederCount()	154
tibasSpaceDef_GetUpdateTransport()	155
tibasSpaceDef_SetUpdateTransport()	156
tibasSpaceDef_SetKey()	157
tibasSpaceDef_SetDistributionFields()	158
tibasSpaceDef_GetFieldDef()	159
tibasSpaceDef_GetDistributionFields()	160
tibasSpaceDef_PutFieldDef()	161
tibasSpaceDef_GetFieldDefs()	162
tibasSpaceDef_TakeFieldDef()	163
tibasSpaceDef_SetKeyDef()	164
tibasSpaceDef_GetKeyDef()	165
tibasSpaceDef_AddIndexDef()	166
tibasSpaceDef_GetIndexDef()	167
tibasSpaceDef_GetIndexDefs()	168
tibasSpaceDef_IsSyncReplicated()	169
tibasSpaceDef_SetSyncReplicated()	170
tibasSpaceDef_GetDistributionPolicy()	171
tibasSpaceDef_SetDistributionPolicy()	172
tibasSpaceDef_GetSpaceWait()	173
tibasSpaceDef_SetSpaceWait()	174
tibasSpaceDef_GetWriteTimeout()	175
tibasSpaceDef_SetWriteTimeout()	176
tibasSpaceDef_GetReadTimeout()	177
tibasSpaceDef_SetReadTimeout()	178
tibasSpaceDef_GetTTL()	179
tibasSpaceDef_SetTTL()	180
tibasSpaceDef_GetLockTTL()	181
tibasSpaceDef_SetLockTTL()	182
tibasSpaceDef_GetLockWait()	183
tibasSpaceDef_SetLockWait()	184
tibasSpaceDef_SetPersisted()	185
tibasSpaceDef_IsPersisted()	186
tibasSpaceDef_SetPersistenceType()	187
tibasSpaceDef_GetPersistenceType()	188

tibasSpaceDef_Free()	189
Chapter 6 MemberDef	191
MemberDef Operations	192
tibasMemberDef_Create()	194
tibasMemberDef_Free()	195
tibasMemberDef_SetMemberName()	196
tibasMemberDef_SetDataStore()	197
tibasMemberDef_SetWorkerThreadCount()	198
tibasMemberDef_GetWorkerThreadCount()	199
tibasMemberDef_GetMemberName()	200
tibasMemberDef_GetDataStore()	201
tibasMemberDef_SetRemoteDiscovery()	202
tibasMemberDef_GetRemoteDiscovery()	203
tibasMemberDef_SetRemoteListen()	204
tibasMemberDef_SetListen()	205
tibasMemberDef_SetDiscovery()	206
tibasMemberDef_GetListen()	208
tibasMemberDef_GetDiscovery()	209
tibasMemberDef_SetContext()	210
tibasMemberDef_GetSecurityPolicyFile()	211
tibasMemberDef_GetSecurityTokenFile()	212
tibasMemberDef_GetAuthenticationCallback()	213
tibasMemberDef_SetSecurityPolicyFile()	214
tibasMemberDef_SetSecurityTokenFile()	215
tibasMemberDef_SetAuthenticationCallback()	216
tibasMemberDef_SetTimeout()	219
Chapter 7 FieldDef	221
FieldDef Operations	222
tibasFieldDef_Create()	223
tibasFieldDef_CreateEx()	225
tibasFieldDef_GetName()	226
tibasFieldDef_GetType()	227
tibasFieldDef_IsNullable()	228
tibasFieldDef_SetNullable()	229
tibasFieldDef_SetEncrypted()	230
tibasFieldDef_IsEncrypted()	231
tibasFieldDef_Free()	232
Chapter 8 FieldDefList	233
FieldDefList Operations	234
tibasFieldDefList_Size()	235

tibasFieldDefList_Get()

tibasFieldDefList_Free().

Chapter 9 KeyDef

KeyDef Operations

tibasKeyDef_Create().

tibasKeyDef_GetIndexType().

tibasKeyDef_SetIndexType().

tibasKeyDef_GetFieldNames().

tibasKeyDef_SetFieldNames().

tibasKeyDef_Free().

Chapter 10 IndexDef.

IndexDef Operations

tibasIndexDef_Create().

tibasIndexDef_GetName().

tibasIndexDef_SetName().

tibasIndexDef_GetIndexType().

tibasIndexDef_SetIndexType().

tibasIndexDef_GetFieldNames().

tibasIndexDef_SetFieldNames().

tibasIndexDef_Free().

tibasIndexDefList_Size().

tibasIndexDefList_Get().

tibasIndexDefList_Free().

Chapter 11 BrowserDef

BrowserDef Operations.

tibasBrowserDef_Create().

tibasBrowserDef_CreateEx().

tibasBrowserDef_GetDistributionScope().

tibasBrowserDef_GetTimeout().

tibasBrowserDef_GetTimeScope().

tibasBrowserDef_SetDistributionScope().

tibasBrowserDef_SetTimeScope().

tibasBrowserDef_SetTimeout().

tibasBrowserDef_Free().

Chapter 12 Browser

Browser Operations

tibasBrowser_Next().

tibasBrowser_Free().

236

237

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

261

262

263

264

266

267

268

269

270

271

272

273

274

275

276

TIBCO ActiveSpaces C Reference

Chapter 13 EventBrowserDef	277
EventBrowserDef Operations	278
tibasEventBrowserDef_Create()	279
tibasEventBrowserDef_CreateEx()	280
tibasEventBrowserDef_GetDistributionScope()	282
tibasEventBrowserDef_GetTimeout()	283
tibasEventBrowserDef_GetTimeScope()	284
tibasEventBrowserDef_SetDistributionScope()	285
tibasEventBrowserDef_SetTimeScope()	286
tibasEventBrowserDef_SetTimeout()	287
tibasEventBrowserDef_Free()	288
Chapter 14 EventBrowser	289
EventBrowser Operations	290
tibasEventBrowser_Free()	291
tibasEventBrowser_Next()	292
Chapter 15 ListenerDef	293
ListenerDef Operations	294
tibasListenerDef_Create()	295
tibasListenerDef_CreateEx()	296
tibasListenerDef_GetTimeScope()	298
tibasListenerDef_GetDistributionScope()	299
tibasListenerDef_SetTimeScope()	300
tibasListenerDef_SetDistributionScope()	301
tibasListenerDef_Free()	302
Chapter 16 Listener	303
Listener Operations	304
tibasListener_Create()	305
tibasSpaceMemberListener_Create()	307
tibasMemberListener_Create()	308
tibasSpaceRemoteMemberListener_Create()	309
tibasRemoteMemberListener_Create()	310
tibasSpaceStateListener_Create()	311
tibasSpaceDefListener_Create()	312
Chapter 17 SpaceEvent, SpaceMemberEvent, SpaceRemoteMemberEvent, and RemoteMemberEvent	313
Space Event Operations	314
tibasSpaceEvent_GetTuple()	316
tibasSpaceEvent_GetType()	317

tibasSpaceEvent_GetSpace()	318
tibasSpaceEvent_Free()	319
tibasSpaceEvent_HasOldTuple()	320
tibasSpaceEvent_GetOldTuple()	321
tibasSpaceEvent_GetMetaspaceName()	322
tibasSpaceEvent_GetSpaceName()	323
tibasSpaceMemberEvent_GetType()	324
tibasSpaceMemberEvent_GetSpaceName()	325
tibasSpaceMemberEvent_GetDistributionRole()	326
tibasSpaceMemberEvent_GetMember()	327
tibasSpaceMemberEvent_Free()	328
tibasMemberEvent_GetType()	329
tibasMemberEvent_GetManagementRole()	330
tibasMemberEvent_GetMember()	331
tibasMemberEvent_Free()	332
tibasSpaceRemoteMemberEvent_GetType()	333
tibasSpaceRemoteMemberEvent_GetSpaceName()	334
tibasSpaceRemoteMemberEvent_GetRemoteMember()	335
tibasSpaceRemoteMemberEvent_GetProxyMember()	336
tibasSpaceRemoteMemberEvent_Free()	337
tibasRemoteMemberEvent_GetType()	338
tibasRemoteMemberEvent_GetRemoteMember()	339
tibasRemoteMemberEvent_GetProxyMember()	340
tibasRemoteMemberEvent_Free()	341

Chapter 18 Tuple..... 343

Tuple Operations.....	344
tibasTuple_Create()	347
tibasTuple_Clone()	348
tibasTuple_GetBoolean()	349
tibasTuple_GetChar()	350
tibasTuple_GetShort()	351
tibasTuple_GetInt()	352
tibasTuple_GetLong()	353
tibasTuple_GetFloat()	354
tibasTuple_GetDouble()	355
tibasTuple_GetString()	356
tibasTuple_GetDateTime()	357
tibasTuple_GetBlob()	358
tibasTuple_PutBoolean()	359
tibasTuple_PutChar()	360
tibasTuple_PutShort()	361
tibasTuple_PutInt()	362
tibasTuple_PutLong()	363

tibasTuple_PutFloat()	364
tibasTuple_PutDouble()	365
tibasTuple_PutString()	366
tibasTuple_PutDateTime()	367
tibasTuple_PutBlob()	369
tibasTuple_Remove()	370
tibasTuple_Size()	371
tibasTuple_Clear()	372
tibasTuple_ToString()	373
tibasTuple_Serialize()	374
tibasTuple_Deserialize()	375
tibasTuple_IsNull()	376
tibasTuple_Exists()	377
tibasTuple_GetFieldNames()	378
tibasTuple_GetFieldType()	379
tibasTuple_PutAll()	380
tibasTuple_Eval()	381
tibasTuple_Free()	382
Chapter 19 TupleList	383
TupleList Operations	384
tibasTupleList_Create()	385
tibasTupleList_Size()	386
tibasTupleList_Get()	387
tibasTupleList_Put()	388
tibasTupleList_Free()	389
Chapter 20 SpaceResult and InvokeResult	391
SpaceResult and InvokeResult Operations	392
tibasSpaceResult_GetTuple()	393
tibasSpaceResult_GetStatus()	394
tibasSpaceResult_GetError()	395
tibasSpaceResult_HasError()	396
tibasSpaceResult_Free()	397
tibasInvokeResult_GetStatus()	398
tibasInvokeResult_GetReturn()	399
tibasInvokeResult_GetMember()	400
tibasInvokeResult_HasError()	401
tibasInvokeResult_GetError()	402
tibasInvokeResult_Free()	403
Chapter 21 SpaceResultList and InvokeResultList	405
SpaceResultList and InvokeResultList Operations	406

tibasSpaceResultList_HasError()	408
tibasResultList_GetTuple()	409
tibasSpaceResultList_GetTuples()	410
tibasSpaceResultList_Size()	411
tibasSpaceResultList_Get()	412
tibasSpaceResultList_GetStatus()	413
tibasSpaceResultList_GetError()	414
tibasSpaceResultList_Put()	415
tibasSpaceResultList_Free()	416
tibasInvokeResultList_Put()	417
tibasInvokeResultList_Size()	418
tibasInvokeResultList_Get()	419
tibasInvokeResultList_GetStatus()	420
tibasInvokeResultList_GetReturn()	421
tibasInvokeResultList_GetReturns()	422
tibasInvokeResultList_GetError()	423
tibasInvokeResultList_HasError()	424
tibasInvokeResultList_Free()	425
Chapter 22 Filter	427
Filter Operations	428
tibasFilter_Create()	429
tibasFilter_Eval()	430
tibasFilter_Free()	431
Chapter 23 Member	433
Member Operations	434
tibasMember_Free()	435
tibasMember_GetName()	436
tibasMember_GetManagementRole()	437
tibasMember_GetDistributionRole()	438
tibasMember_GetContext()	439
tibasMember_GetHostAddress()	440
tibasMember_GetPort()	441
tibasMember_GetJoinTime()	442
tibasMember_GetId()	443
Chapter 24 MemberList	445
MemberList Operations	446
tibasMemberList_Size()	447
tibasMemberList_Get()	448
tibasMemberList_Free()	449

Chapter 25 Persister, Action, Op, OpList, ActionResult, LogLevel, and Recovery Functions 451

Persister Operations	452
tibasPersister_Create()	453
tibasPersister_Free()	455
Action, Op, OpList, ActionResult, Invocable, and LogFile Operations	456
tibasAction_GetType()	459
tibasAction_GetTuple()	460
tibasAction_GetSpace()	461
tibasAction_GetSpaceName()	462
tibasAction_GetOps()	463
tibasAction_Free()	464
tibasOp_GetType()	465
tibasOp_GetTuple()	466
tibasOp_GetOldTuple()	467
tibasOp_HasOldTuple()	468
tibasOp_Free()	469
tibasOpList_Size()	470
tibasOpList_Get()	471
tibasOpList_Free()	472
tibasActionResult_SetFailed()	473
tibasActionResult_GetStatus()	474
tibasActionResult_SetTuple()	475
tibasActionResult_GetTuple()	476
tibas_SetInvocable()	477
tibas_SetMemberInvocable()	479
tibas_SetLogLevel()	481
tibas_GetLogLevel()	482
tibas_EnableFileLogging()	483
tibas_DisableFileLogging()	484

Chapter 26 StringList..... 485

StringList Operations	486
tibasStringList_Free()	487
tibasStringList_Get()	488
tibasStringList_Size()	489

Chapter 27 Error..... 491

Error Operations	492
tibasError_GetError()	493
tibasError_GetSevereError()	494
tibasError_GetCode()	495
tibasError_GetMessage()	496

tibasError_GetStackTrace()	497
tibasError_Free()	498
Appendix A Error Codes	499
Error Codes	500
Appendix B Datatypes	505
C Datatypes	506
Enumerated Types	507
tibas_boolean	509
tibas_loglevel	510
tibas_status	511
tibas_type	512
tibas_indexType	513
tibas_distributionPolicy	514
tibas_evictionPolicy	515
tibas_distributionRole	516
tibas_persistenceType	517
tibas_persistencePolicy	518
tibas_managementRole	519
tibas_eventType	520
tibas_memberEventType	521
tibas_spaceState	522
tibas_lockType	523
tibas_lockScope	524
tibas_browserType	525
tibas_distributionScope	526
tibas_updateTransport	527
tibas_timeScope	528
tibas_actionType	529
tibas_opType	530
tibas_authenticationMethod	531
Index	533

Preface

TIBCO ActiveSpaces® is a distributed peer-to-peer in-memory data grid, a form of virtual shared memory that leverages a distributed hash table with configurable replication.

TIBCO ActiveSpaces combines the features and performance of databases, caching systems, and messaging software to support large, highly volatile data sets and event-driven applications. It lets you off-load transaction-heavy systems and allows developers to concentrate on business logic rather than the complexities of developing distributed fault-tolerance.

TIBCO ActiveSpaces is available in three versions:

- **TIBCO ActiveSpaces® Enterprise Edition**—Provides C, Java, and .NET API sets and enables full cluster functionality. To enable remote clients, you must purchase licenses for the TIBCO ActiveSpaces Remote Client Edition.
- **TIBCO ActiveSpaces® Remote Client Edition**—Can be purchased in addition to the Enterprise Edition. Allows you to set up remote clients. Applications running on the remote clients can access the data grid and perform most ActiveSpaces operations.
- **TIBCO ActiveSpaces® Developer Edition**—A developer version of the product.. This version is downloadable from TIBCO Developer Network at <http://tap.tibco.com>.

This manual describes the TIBCO ActiveSpaces C API. The manual includes all the ActiveSpaces functions and error codes available to C programmers.

Topics

- [Related Documentation, page xvi](#)
- [Typographical Conventions, page xvii](#)
- [How to Contact TIBCO Support, page xix](#)

Related Documentation

This section lists documentation resources you may find useful.

TIBCO ActiveSpaces Documentation

The following documents form the TIBCO ActiveSpaces documentation set:

- *TIBCO ActiveSpaces Installation* Read this manual for instructions on site preparation and installation.
- *TIBCO ActiveSpaces Administration* Read this manual to gain an understanding of the product that you can apply to the various tasks you may undertake.
- *TIBCO ActiveSpaces Developer's Guide* Read this manual for instructions on using the product to develop an application that manages data grids.
- *TIBCO ActiveSpaces C Reference* Read this manual for reference information on the C functions for developing an application that manages data grids.
- *TIBCO ActiveSpaces Release Notes* Read the release notes for a list of new and changed features. This document also contains lists of known issues and closed issues for this release.

Other TIBCO Product Documentation

You might find it useful to read the documentation for the following TIBCO products:

- TIBCO Rendezvous® software: TIBCO Rendezvous provides an optional transport that you can choose to use in place of the built-in Pragmatic General Multicast (PGM) multicast transport that TIBCO ActiveSpaces uses by default. TIBCO Rendezvous handles the transport of data and messages between member processes over the network.

For information on TIBCO Rendezvous, see *TIBCO Rendezvous Concepts*, Chapter 8 “Transport,” for information about the service, network, and daemon parameters, which are used to configure discovery and listen transport in TIBCO ActiveSpaces.

model-driven approach to collect, filter, and correlate events and deliver real-time operational insight. TIBCO BusinessEvents provides a data grid based on TIBCO ActiveSpaces, but does not provide an API or a user interface allowing direct access to ActiveSpaces functionality.

Typographical Conventions

The following typographical conventions are used in this manual.

Table 1 General Typographical Conventions


Convention	Use
<i>TIBCO_HOME</i>	All TIBCO products are installed under the same directory. This directory is referenced in documentation as <i>TIBCO_HOME</i> . The value of <i>TIBCO_HOME</i> depends on the operating system. For example, on Windows systems, the default value is C:\tibco.
code font	Code font identifies commands, code examples, filenames, path names, and output displayed in a command window. For example: Use MyCommand to start the foo process.
bold code font	Bold code font is used in the following ways: <ul style="list-style-type: none"> In procedures, to indicate what a user types. For example: Type admin. In large code samples, to indicate the parts of the sample that are of particular interest. In command syntax, to indicate the default parameter for a command. For example, if no parameter is specified, MyCommand is enabled: MyCommand [enable disable]
<i>italic font</i>	Italic font is used in the following ways: <ul style="list-style-type: none"> To indicate a document title. For example: See <i>TIBCO ActiveMatrix BusinessWorks Concepts</i>. To introduce new terms. For example: A portal page may contain several portlets. <i>Portlets</i> are mini-applications that run in a portal. To indicate a variable in a command or code syntax that you must replace. For example: MyCommand <i>pathname</i>
Key combinations	Key name separated by a plus sign indicate keys pressed simultaneously. For example: Ctrl+C. Key names separated by a comma and space indicate keys pressed one after the other. For example: Esc, Ctrl+Q.
	The note icon indicates information that is of special interest or importance, for example, an additional action required only in certain circumstances.

Table 1 General Typographical Conventions (Cont'd)



Convention	Use
	The tip icon indicates an idea that could be useful, for example, a way to apply the information provided in the current section to achieve a specific result.
	The warning icon indicates the potential for a damaging situation, for example, data loss or corruption if certain steps are taken or not taken.

Table 2 Syntax Typographical Conventions

Convention	Use
[]	<p>An optional item in a command or code syntax.</p> <p>For example:</p> <pre>MyCommand [optional_parameter] required_parameter</pre>
	<p>A logical OR that separates multiple items of which only one may be chosen.</p> <p>For example, you can select only one of the following parameters:</p> <pre>MyCommand param1 param2 param3</pre>
{ }	<p>A logical group of items in a command. Other syntax notations may appear within each logical group.</p> <p>For example, the following command requires two parameters, which can be either the pair param1 and param2, or the pair param3 and param4.</p> <pre>MyCommand {param1 param2} {param3 param4}</pre> <p>In the next example, the command requires two parameters. The first parameter can be either param1 or param2 and the second can be either param3 or param4:</p> <pre>MyCommand {param1 param2} {param3 param4}</pre> <p>In the next example, the command can accept either two or three parameters. The first parameter must be param1. You can optionally include param2 as the second parameter. And the last parameter is either param3 or param4.</p> <pre>MyCommand param1 [param2] {param3 param4}</pre>

How to Contact TIBCO Support

For comments or problems with this manual or the software it addresses, please contact TIBCO Support as follows.

- For an overview of TIBCO Support, and information about getting started with TIBCO Support, visit this site:

<http://www.tibco.com/services/support>

- If you already have a valid maintenance or support contract, visit this site:

<https://support.tibco.com>

Entry to this site requires a user name and password. If you do not have a user name, you can request one.

Chapter 1 Introduction

TIBCO ActiveSpaces provides API sets in several programming languages that allow developers to utilize TIBCO ActiveSpaces functionality from within their applications. This chapter introduces the TIBCO ActiveSpaces API set for the C programming language.

Topics

- [Overview, page 2](#)
- [C API Header Files, page 3](#)
- [Status Codes, page 4](#)
- [Building Your Application, page 5](#)
- [Running Your Application, page 7](#)

Overview

Prior to trying to develop an application which utilizes TIBCO ActiveSpaces, it is important to understand TIBCO ActiveSpaces as a product, its various components, and the concepts surrounding how TIBCO ActiveSpaces can be utilized. This information can be found in the document *TIBCO ActiveSpaces Developer's Guide*.

TIBCO ActiveSpaces also provides example C programs that exercise the various features of TIBCO ActiveSpaces. These example programs can be found in the directory `AS_HOME/examples/c`. It is recommended that you compile, link and run the examples found in this directory to help you learn and understand how to develop an application using the TIBCO ActiveSpaces C API. For more information on the examples, see Chapter 4, "Using the Example Code," in the *TIBCO ActiveSpaces Developer's Guide*.

Once you become familiar with TIBCO ActiveSpaces high-level usage concepts, by reading the Developer's Guide and running some of the examples, you can then refer to the remainder of this document to find detailed information on the C language API provided by TIBCO ActiveSpaces and how to use the API in your applications.

C API Header Files

The header files containing the function declarations and typedefs that make up the TIBCO ActiveSpaces C API are found in the `AS_HOME/include/c` directory. Each header file is named based upon the TIBCO ActiveSpaces object for which the file contains declarations or definitions. For example, `browser.h` contains the declaration of functions related to the use of Space Browsers while `eventbrowser.h` contains the declaration of functions related to the use of Event Browsers.

A single header file named `tibas.h` is provided, which includes all of the header files that comprise the TIBCO ActiveSpaces C API. The `tibas.h` file also contains the declaration of some functions that do not fit into the other header files. Therefore, it is recommended that you always include `tibas.h` in your application instead of including the separate header files.

Status Codes

Most of the functions in the TIBCO ActiveSpaces C API return a result of type `tibas_status`, which is defined in the header file `types.h`. The value of `tibas_status` is an enum which can have the following values indicated in [Table 3, Status Codes](#):

Table 3 Status Codes

Constant	Type	Description
TIBAS_OK	no error	The operation was successful.
TIBAS_ALREADY_EXISTS	error	The operation failed because updating from null occurred, because there is currently a tuple in the space for the requested key field(s) value(s).
TIBAS_LOCKED	error	The operation failed because the tuple was already locked by another thread or process (depending on the LockScope).
TIBAS_MISMATCHED_LOCK	error	The lock expired in the space and another member already locked the tuple.
TIBAS_INCOMPATIBLE_TUPLE	error	The operation failed because the name of a field contained in the tuple is incompatible with the space definition.
TIBAS_MISMATCHED_TUPLE	error	The operation failed because the tuple is incompatible with the space definition.
TIBAS_INCOMPATIBLE_TYPE	severe error	The operation failed because two data types involved in an operation are incompatible
TIBAS_LIMIT_EXCEEDED	error	The operation failed because a predefined limit on the space, such as capacity, has been exceeded.
TIBAS_INVALID_ARG	severe error	The operation failed because an invalid argument was passed to an operation.
TIBAS_SYS_ERROR	severe error	The operation failed because of a system error.

Building Your Application

The C language examples directory for TIBCO ActiveSpaces includes a Makefile that you can use to compile and link the example programs. It is recommended that you examine this Makefile to see which compiler flags and linker options to use for building your application.

[Table 4, ActiveSpaces Compiler Platforms](#) lists compilers that you can use for each platform supported by TIBCO ActiveSpaces:

Table 4 *ActiveSpaces Compiler Platforms*

Compiler	Platforms
xlC	AIX
aCC	HP-UX
gcc	Linux, Mac
CC	Solaris
cl	Windows

When linking your application, you might need to link the libraries listed in [Table 5, Link Libraries for ActiveSpaces](#), depending on your platform and whether you statically link libraries:

Table 5 *Link Libraries for ActiveSpaces*

Platform	Library	Description
All	as-common	Required for by all applications that use the C API.
All	as-tibpgm	Required for applications that use <code>tibpgm</code> for discovery.
All	as-tibrv	Required for applications that use “ <code>tibrv</code> ” for discovery.
All	tibrv	TIBCO Rendezvous library,. Only needed when <code>tibrv</code> is used for discovery instead of <code>tibpgm</code> .

Table 5 *Link Libraries for ActiveSpaces*

Platform	Library	Description
Linux, Mac, AIX, HP-UX	pthread	Posix threads library.
Solaris	thread	Solaris threads library.
Linux, Mac, AIX	dl	Dynamic loading library.
Windows	ws2_32.lib	Windows socket library.
Windows	advapi32.lib	Windows advanced services library.

Running Your Application

To run your application, ensure that your environment has been set up properly for running a TIBCO ActiveSpaces application. Ensure that your path or library path is set up properly so that the TIBCO ActiveSpaces libraries can be found when you run your application. For information on setting up your environment, see [Setting Environment Variables on page 12](#) of [Chapter 2, Installation Steps](#), in the *TIBCO ActiveSpaces Installation* document.

Chapter 2 Metaspace

A metaspace represents:

- From a deployment perspective, the cluster of ActiveSpaces processes sharing the same metaspace name and discovery URL, making an instance of an ActiveSpaces deployment. The processes in a metaspace provide the resources for hosting spaces.
- From an administrative perspective, a container for a set of spaces. There are two kinds of spaces contained in a metaspace: *system spaces*, which are defined internally by ActiveSpaces, and *user spaces*, which are defined by a user.

The metaspace is created when the first process connects to it, and it disappears when the last process disconnects from it.

For detailed information on metaspaces, see [What is a Metaspace? on page 13](#) of the *TIBCO ActiveSpaces Developer's Guide*.

Topics

- [Metaspace Operations](#)

Metaspace Operations

The following table lists the Metaspace operations:

Table 6 *Metaspace*

Function or Type	Description	Page
<code>tibasMetaspace_Connect()</code>	Establishes a connection to a metaspace and returns a new <code>Metaspace</code> object.	14
<code>tibasMetaspace_ConnectEx()</code>	Establishes a connection to a metaspace and returns a new <code>Metaspace</code> object.	15
<code>tibasMetaspace_GetSpace()</code>	Attempts to join a metaspace or get a reference to a currently joined metaspace. Returns a <code>Space</code> object.	18
<code>tibas_GetMetaspace()</code>	Creates a metaspace and assigns it a specified name.	16
<code>tibas_GetMetaspaceNames()</code>	Returns a list of the currently defined metaspaces.	17
<code>tibasMetaspace_GetSpaceEx()</code>	The function accepts a <code>SpaceMemberDef</code> object that you can use to specify whether the space member is a seeder or a leech, and if you are implementing shared-nothing persistence, to specify a directory where persisted data is stored.	20
<code>tibasMetaspace_GetSystemSpace()</code>	Returns a valid <code>Space</code> object for one of the system spaces. System spaces are read-only and have a unique distribution role.	21
<code>tibasMetaspace_GetUserSpaceNames()</code>	Returns a list of strings containing the name of all the spaces defined in the metaspace.	22
<code>tibasMetaspace_BeginTransaction()</code>	Starts a transaction for the current thread.	43
<code>tibasMetaspace_GetRemoteListen()</code>	Returns the listen URL for a specified metaspace.	38
<code>tibasMetaspace_Browse()</code>	Creates a space browser on the specified space.	39

Table 6 *Metaspace*

Function or Type	Description	Page
<code>tibasMetaspace_BrowseEvents()</code>	Creates an event browser on the specified space. May cause the space to be automatically joined with a distribution role of <code>TIBAS_DISTRIBUTION_ROLE_LEECH</code> if not already joined.	41
<code>tibasMetaspace_CommitTransaction()</code>	Commits all the space operations invoked since the <code>tibasMetaspace_BeginTransaction()</code> function or the <code>tibasMetaspace_AcquireContext()</code> function was called.	44
<code>tibasMetaspace_DefineSpace()</code>	Defines a user space.	23
<code>tibasMetaspace_DropSpace()</code>	Deletes an existing space definition from a specified metaspace.	25
<code>tibasMetaspace_AlterSpace()</code>	Alters a specified space definition for a specified metaspace.	26
<code>tibasMetaspace_Free()</code>	Frees resources used by the Metaspace object.	57
<code>tibasMetaspace_GetLogLevel()</code>	Returns the log level set for the metaspace.	52
<code>tibasMetaspace_GetRemoteMembers()</code>	Returns a list of the remote members of the metaspace.	48
<code>tibasMetaspace_GetDiscovery()</code>	Returns the discovery URL for the metaspace connection.	53
<code>tibasMetaspace_GetRemoteDiscovery()</code>	Gets the remote discovery URL for a specified metaspace.	55
<code>tibasMetaspace_GetSpaceMembers()</code>	Returns the member for the current metaspace connection.	50
<code>tibasMetaspace_GetSelfMember()</code>	Returns the member for the current metaspace connection.	51
<code>tibasMetaspace_RecoverSpace()</code>	Restarts a space that is suspended. Recovers data from persistence files.	27
<code>tibasMetaspace_RecoverSpaceEx()</code>	Restarts a space that is suspended. Recovers data from persistence files.	28

Table 6 *Metaspace*

Function or Type	Description	Page
<code>tibasMetaspace_GetSpaceDef()</code>	Returns the space definition for a specified space name.	29
<code>tibasMetaspace_GetListen()</code>	Returns the listen attribute defined for a specified metaspace connection.	54
<code>tibasMetaspace_Listen()</code>	Attaches a listener to a space for receiving space events.	30
<code>tibasMetaspace_ListenSpaceMemberEvents()</code>	Attaches a listener to a metaspace for receiving metaspace member events.	33
<code>tibasMetaspace_ListenMemberEvents()</code>	Attaches a listener to a space for receiving space member events.	34
<code>tibasMetaspace_ListenSpaceRemoteMemberEvents()</code>	Attaches a listener to a metaspace for receiving remote space member events.	35
<code>tibasMetaspace_ListenSpaceState()</code>	Attaches a listener to a metaspace for receiving space state events.	36
<code>tibasMetaspace_ListenSpaceDef()</code>	Attaches a listener to a metaspace for receiving space definition events.	37
<code>tibasMetaspace_ReleaseContext()</code>	Releases the thread context from a specified thread that called <code>tibasMetaspace_BeginTransaction()</code> .	46
<code>tibasMetaspace_RollbackTransaction()</code>	Rolls back all of the space operations invoked since the <code>tibasMetaspace_BeginTransaction()</code> function or the <code>tibasMetaspace_AcquireContext()</code> function was called.	45
<code>tibasMetaspace_AcquireContext()</code>	Acquires a thread context that has been released by another thread through a call to the <code>tibasMetaspace_ReleaseContext()</code> function.	47
<code>tibasMetaspace_GetRemoteMembers()</code>	Returns a list of the remote members of the metaspace.	48
<code>tibasMetaspace_SetLogLevel()</code>	Sets the log level for the metaspace.	56

Table 6 *Metaspace*

Function or Type	Description	Page
Data		
tibas_FreeData()	Frees resources given to the user.	58

tibasMetaspace_Connect()

Function

Declaration `tibas_status tibasMetaspace_Connect`
 `(tibasMetaspace* metaspace,`
 `const char* metaspaceName,`
 `tibasMemberDef memberDef)`

Purpose Establishes a connection to a metaspace and returns a new Metaspace object.

Parameters	Parameter	Description
	<code>metaspace</code>	Returns a Metaspace object that you use to define, join, or leave spaces.
	<code>metaspaceName</code>	A string specifying the metaspace name. If null or "" is passed, the default name <code>ms</code> is used.
	<code>memberDef</code>	A MemberDef object used to specify member and connection attributes.

Remarks Use the `tibasMetaspace_Connect()` function to connect to a specified metaspace and specify member and connection attributes for the connection, such as the discovery URL used to discover other members of the metaspace and the TCP port used to listen for incoming connections from new members.

Before calling `tibasMetaspace_Connect()`, create a MemberDef object and specify the attributes for the member definition using the `tibasMemberDef` functions. The `memberDef` parameter specifies a MemberDef struct that defines the characteristics of the connection, such as the discovery attribute and the listen attribute used for connections to the metaspace.

See Also [tibasMemberDef_Create\(\)](#), [tibasMemberDef_SetListen\(\)](#),
[tibasMemberDef_SetDiscovery\(\)](#), [tibasMemberDef_SetRemoteDiscovery\(\)](#)

tibasMetaspace_ConnectEx()

Function

Declaration `tibas_status` tibasMetaspace_ConnectEx
 (`tibasMetaspace*` metaspace,
 `const char*` args)

Purpose Establishes a connection to a metaspace and returns a new Metaspace object.

Parameters	Parameter	Description
	metaspace	Returns a Metaspace object that you use to define, join, or leave spaces.
	args	A string specifying additional arguments specific to your application.

Remarks Use the `tibasMetaspace_ConnectEx()` function to connect to a specified metaspace and specify additional arguments specific to your application.

See Also [tibasMetaspace_Connect\(\)](#)

tibas_GetMetaspace()

Function

Declaration `tibas_status` `tibas_GetMetaspace`(
 `tibasMetaspace*` `metaspace`,
 `const char*` `metaspaceName`)

Purpose Creates a metaspace and assigns it a specified name.

Parameters	Parameter	Description
	<code>metaspace</code>	Pointer to a <code>tibasMetaspace</code> struct that returns the metaspace definition.
	<code>metaspaceName</code>	Pointer to a string specifying the metaspace name.

Remarks Use the `tibas_GetMetaspace()` function to create a metaspace with a specified name.

You can only use `tibas_GetMetaspace()` if your application already has a connection to a metaspace, which would normally be instantiated by calling `tibasMetaspaceConnect()`. The `tibas_GetMetaspace` function() is provided as a convenience function, which you can call if your application needs a second connection to a metaspace.

The function returns a Metaspace object that you can use to connect to the metaspace.

See Also [tibasMetaspace_Connect\(\)](#)

tibas_GetMetaspaceNames()

Function

Declaration `tibas_status tibas_GetMetaspaceNames(
 tibasStringList* metaspaceNames)`

Purpose Returns a list of the currently defined metaspaces.

Parameters

Parameter	Description
metaspaceNames	Returns a list of the currently defined metaspaces.

Remarks Use the `tibas_GetMetaspaceNames()` function to return a list of the currently defined metaspaces.

tibasMetaspace_GetSpace()

Function

Declaration

```
tibas_status tibasMetaspace_GetSpace(  
    tibasMetaspace    metaspace,  
    tibasSpace*       space,  
    const char*       spaceName,  
    tibas_distributionRole distributionRole)
```

Purpose

Attempts to join a metaspace or get a reference to a currently joined metaspace. Returns a Space object.

Parameters

Parameter	Description
metaspace	Specifies a Metaspace object that identifies the metaspace to join.
space	The returned Space object.
spaceName	The name to be assigned to the space when the metaspace is joined.
distributionRole	The distribution role to join the space as. Can be one of the following: <ul style="list-style-type: none">TIBAS_DISTRIBUTION_ROLE_LEECHTIBAS_DISTRIBUTION_ROLE_SEEDERTIBAS_DISTRIBUTION_ROLE_NONE

Remarks

Use the `tibasMetaspace_GetSpace()` function to join a specified space and specify a distribution role for the space member.

The distribution role can be one of the following:

- TIBAS_DISTRIBUTION_ROLE_SEEDER** Specifies that the space member is a seeder—an application that participates in the storing of data in the space, and can read and write data. When seeder applications join or leave the space, ActiveSpaces redistributes the data in the space as necessary to maintain even data distribution.
- TIBAS_DISTRIBUTION_ROLE_LEECH** Specifies that the space member is a leech—an application that passively participates in the space and does not read and write data or cause redistribution of space data when it joins or leaves the space.
- TIBAS_DISTRIBUTION_ROLE_NONE** (default setting)

See Also [tibasMetaspace_GetSpaceEx\(\)](#)

tibasMetaspace_GetSpaceEx()

Function

Declaration

```
tibas_status tibasMetaspace_GetSpaceEx(  
    tibasMetaspace metaspace,  
    tibasSpace* space,  
    const char* spaceName,  
    tibasSpaceMemberDef spaceMemberDef)
```

Purpose

Attempts to join a space or get a reference to a currently joined space. Returns a Space object.

Parameters

Parameter	Description
metaspace	The TIBCO ActiveSpaces entity on which the function is invoked.
space	The returned Space object.
spaceName	The name of the space to join
spaceMemberDef	Specifies a SpaceMemberDef object to specify optional parameters.

Remarks

Use the `tibasMetaspace_GetSpaceEx()` function to join a space and specify connection attributes for the space member.

The function accepts a `SpaceMemberDef` object that you can use to specify whether the space member is a seeder or a leech, and if you are implementing shared-nothing persistence, to specify a directory where persisted data is stored.

tibasMetaspace_GetSystemSpace()

Function

Declaration `tibas_status` tibasMetaspace_GetSystemSpace(
 tibasMetaspace metaspace,
 tibasSpace* space,
 const char* spaceName)

Purpose Returns a valid Space object for one of the system spaces. System spaces are read-only and have a unique distribution role.

Parameters	Parameter	Description
	metaspace	The TIBCO ActiveSpaces entity on which the function is invoked.
	space	The returned Space object.
	spaceName	The name of the system space.

Remarks Use the `tibasMetaspace_GetSystemSpace()` function to return a Space object showing system space information for a specified system space.

tibasMetaspace_GetUserSpaceNames()

Function

Declaration `tibas_status tibasMetaspace_GetUserSpaceNames(
 tibasMetaspace metaspace,
 tibasStringList* spaceNameList)`

Purpose Returns a list of strings containing the name of all the spaces defined in the metaspace.

Parameters	Parameter	Description
	metaspace	The TIBCO ActiveSpaces entity on which the function is invoked.
	spaceNameList	The returned list of strings.

Remarks Use the `tibasMetaspace_GetUserSpaceNames()` function to return a list of the spaces defined in a specified metaspace.

tibasMetaspace_DefineSpace()

Function

```
Declaration      tibas_status tibasMetaspace_DefineSpace
                   (tibasMetaspace metaspace,
                   tibasSpaceDef   spaceDef)
```

Purpose	Defines a user space.
----------------	-----------------------

Parameters

Parameter	Description
metaspace	A Metaspace object that specifies the metaspace in which to define a user space.
spaceDef	Specify a SpaceDef object that contains attributes used to define the space.

Remarks	Use the <code>tibasMetaspace_DefineSpace()</code> function to define a user space in a specified metaspace.
----------------	---

Before you call `tibasMetaspaceDefineSpace()`:

- Create a space definition by calling the `tibasSpaceDefCreate()` function.
The `tibasSpaceDefCreate()` function returns a `SpaceDef` object that you use as input to `tibasSpaceDef` functions that set attributes of the space and to the `tibasMetaspace_DefineSpace()` function.
- Specify the fields and attributes for the space using the `tibasSpaceDef` functions.



Calling `tibasMetaspace_DefineSpace()` does not automatically join the space. To join the space, call the `tibasMetaspaceGetSpace()` function or the `tibasMetaspaceGetSpaceEx()` function and join the space as a seeder or as a leech.

If the space definition is different than what is already defined, the function returns `TIBAS_SYS_ERROR`. If the definition is same as for a previously created space, the function returns `TIBAS_OK`.

See Also [tibasSpaceDef_Create\(\)](#), [tibasSpaceDef_SetName\(\)](#), [tibasSpaceDef_SetReplicationCount\(\)](#), [tibasSpaceDef_SetCapacity\(\)](#), [tibasSpaceDef_SetEvictionPolicy\(\)](#), [tibasSpaceDef_SetMinSeederCount\(\)](#), [tibasSpaceDef_SetUpdateTransport\(\)](#), [tibasSpaceDef_SetKey\(\)](#), [tibasSpaceDef_PutFieldDef\(\)](#), [tibasSpaceDef_AddIndexDef\(\)](#),

```
tibasSpaceDef_SetSyncReplicated(), tibasSpaceDef_SetDistributionPolicy(),  
tibasSpaceDef_SetTTL(), tibasSpaceDef_SetLockTTL(),  
tibasSpaceDef_SetLockWait(), tibasSpaceDef_SetPersistenceType().,  
tibasMetaspace_GetSpaceEx()
```

tibasMetaspace_DropSpace()

Function

Declaration

```
tibas_status tibasMetaspace_DropSpace
(tibasMetaspace metaspace,
const char*      spaceName)
```

Purpose

Deletes an existing space definition from a specified metaspace.

Parameters

Parameter	Description
metaspace	The TIBCO ActiveSpaces entity on which the function is invoked.
spaceName	The name of the space to drop.

Remarks

Use the `tibasMetaspace_DropSpace()` function to delete a specified space definition from a specified space.

The function returns `SPACE_NAME_INVALID` if the specified space name does not exist.



The space definition cannot be deleted if there are members connected to the space.

tibasMetaspace_AlterSpace()

Function

Declaration `tibas_status tibasMetaspace_AlterSpace(
 tibasMetaspace metaspace,
 tibasSpaceDef spaceDef)`

Purpose Alters a specified space definition for a specified metaspace.

Parameters	Parameter	Description
	metaspace	Specify the <code>tibasMetaspace</code> object that identifies the metaspace that the <code>spaceDef</code> you want to change is associated with.
	spaceDef	Specifies the <code>spaceDef</code> object that you want to modify, or if you are adding a new space definition, specifies the new <code>spaceDef</code> .

Remarks Use the `tibasMetaspace_AlterSpace()` function to alter a space definition for a specified space. You can add new fields to the space definition; however, any new fields that you specify must be set as nullable.

When you specify the `spaceDef` object to pass to the function, you can specify an existing `spaceDef` or specify a new `spaceDef`.

If the `spaceDef` that you pass to the function is not defined, then `tibasMetaspace_AlterSpace()` defines it. If there is an existing `spaceDef`, then the function attempts to modify the existing one, and if the new space definition is not compatible with the existing one, an exception is generated indicating the cause of the error.

If you are you are calling `tibasMetaspace_AlterSpace()` to modify an existing `spaceDef`, then you can use the `tibas_spaceDef` functions to set up the `spaceDef` and the `tibasFieldDef` functions to define the field definitions for the space. Then pass the `spaceDef` object that you have created to `tibasMetaspace_AlterSpace()`.

See Also `tibasSpaceDef_Create()`, `tibasSpaceDef_SetName()`,
`tibasSpaceDef_SetReplicationCount()`, `tibasSpaceDef_SetCapacity()`,
`tibasSpaceDef_SetEvictionPolicy()`, `tibasSpaceDef_SetLockScope()`,
`tibasSpaceDef_SetKey()`, `tibasFieldDef_Create()`, `tibasFieldDef_SetNullable()`

tibasMetaspace_RecoverSpace()

Function

Declaration `tibas_status` tibasMetaspace_RecoverSpace
 (`tibasMetaspace` metaspace,
 const char* spaceName)

Purpose Restarts a space that is suspended. Recovers data from persistence files.

Parameters	Parameter	Description
	metaspace	Specifies the metaspace to which the space belongs.
	spaceName	The space name for the space that is to be recovered.

Remarks Use the `tibasMetaspace_RecoverSpace()` function to restart a space that is in the SUSPENDED state and recover data from persistence files.

 This is useful if the space has lost data due to the system going down or during system maintenance.

 If the specified space name is incorrect, the function returns `SPACE_NAME_INVALID`.

tibasMetaspace_RecoverSpaceEx()

Function

Declaration `tibas_status` tibasMetaspace_RecoverSpaceEx(
 `tibasMetaspace` `metaspace`,
 `const char*` `spaceName`,
 `tibasRecoveryOptions` `recoveryOptions`)

Purpose Restarts a space that is suspended. Optionally recovers data from persistence files.

Parameters	Parameter	Description
	<code>metaspace</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>spaceName</code>	The space name used by the function.
	<code>recoveryOptions</code>	<p>This argument is a boolean value.</p> <p>The default value (<code>TIBAS_TRUE</code>) specifies recovery with data; if you specify <code>TIBAS_FALSE</code>, recovery without data is set.</p> <p>If you specify recovery with data, data is read from disk when the space is recovered. If you specify recovery without data, the space is recovered, but data is not read from disk.</p>

Remarks Use the `tibasMetaspace_RecoverSpaceEx()` function to restart a space that is in the `SUSPENDED` state and specify options for data recovery.

This is useful if the space has lost data due to the system going down or during system maintenance.

Unlike the `tibasMetaspace_RecoverSpace()` function, (`tibasMetaspace_RecoverSpaceEx()` specifies a recovery option. Recovery is possible if the space has been set up with persistence set to shared-all persistence.

If the specified space name is incorrect, the function returns `SPACE_NAME_INVALID`.

See Also `tibasMetaspace_RecoverSpace()`

tibasMetaspace_GetSpaceDef()

Function

Declaration `tibas_status tibasMetaspace_GetSpaceDef
(tibasMetaspace metaspace,
tibasSpaceDef* spaceDef,
const char* spaceName)`

Purpose Returns the space definition for a specified space name.

Parameters

Parameter	Description
metaspace	The TIBCO ActiveSpaces entity on which the function is invoked.
spaceDef	The space definition.
spaceName	The space name.

Remarks Use the `tibasMetaspace_GetSpaceDef()` function to return the `SpaceDef` object for a specified space.

After you have obtained the `SpaceDef` object, you can use it as input to `tibasSpaceDef` functions that set or return various attributes of the space definition.

See Also `tibasMetaspace_DefineSpace()`, `tibasSpaceDef_GetName()`,
`tibasSpaceDef_GetReplicationCount()`, `tibasSpaceDef_GetCapacity()`,
`tibasSpaceDef_GetEvictionPolicy()`, `tibasSpaceDef_GetNumFields()`,
`tibasSpaceDef_GetMinSeederCount()`,
`tibasSpaceDef_GetUpdateTransport()`,
`tibasSpaceDef_IsSyncReplicated()`,
`tibasSpaceDef_GetDistributionPolicy()`, `tibasSpaceDef_GetTTL()`,
`tibasSpaceDef_IsPersisted()`, `tibasSpaceDef_GetLockWait()`,
`tibasSpaceDef_GetLockWait()`, `tibasSpaceDef_IsPersisted()`,
`tibasSpaceDef_GetPersistenceType()`,

tibasMetaspace_Listen()

Function

Declaration

```
tibas_status tibasMetaspace_Listen
(tibasMetaspace metaspac,
const char*      spaceName,
tibasListener    listener,
tibasListenerDef listenerDef,
const char*      filter)
```

Purpose

Attaches a listener to a space for receiving space events.

Parameters

Parameter	Description
metaspac	The TIBCO ActiveSpaces entity on which the function is invoked.
spaceName	The name of the space on which to attach the listener.
listener	The listener.
listenerDef	The listener definition.
filter	A filter string to apply to the listener or NULL.

Remarks

Use the `tibasMetaspace_Listen()` function to attach a listener to a specified space in a specified metaspac. You do not have to be joined to the space to call the function.

Before you can attach a listener to a space, there must be:

- A listener object created by a call to the `tibasListener_Create()` function.
Specify this object with the `listener` parameter.
- A `ListenerDef` object that specifies the characteristics of the listener.
Specify this object with the `listenerDef` parameter.

Calling `tibasMetaspace_Listen()` may cause the space to be automatically joined with a distribution role of `TIBAS_DISTRIBUTION_ROLE_LEECH` if not already joined.

Prior to calling `tibasMetaspace_Listen()`, you must create a `tibasListenerDef` object by calling the `tibasListenerDef_Create()` function or the `tibasListenerDef_CreateEx()` function and specify the attributes of the listener, such as the time scope and distribution scope for the listener.

See Also [tibasListener_Create\(\)](#), [tibasListenerDef_Create\(\)](#), [tibasListenerDef_CreateEx\(\)](#),
[tibasListenerDef_SetTimeScope\(\)](#), [tibasListenerDef_SetDistributionScope\(\)](#),
[tibasSpace_Listen\(\)](#)

tibasMetaspace_ListenSpaceMemberEvents()

Function

Declaration `tibas_status` `tibasMetaspace_ListenSpaceMemberEvents`
 (`tibasMetaspace` `metaspace`,
 `const char*` `spaceName`,
 `tibasListener` `listener`)

Purpose Attaches a listener to a space for receiving space member events.

Parameters	Parameter	Description
	<code>metaspace</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>spaceName</code>	The name of the space on which to attach the listener.
	<code>listener</code>	The listener.

Remarks Use the `tibasMetaspace_ListenSpaceMemberEvents()` function to attach a listener to a specified space member in a specified metaspace. You do not have to be joined to the space to call the function.

Before you can attach a listener to a space, there must be a listener object created by a call to the `tibasListener_Create()` function.

Specify this object with the `listener` parameter.

See Also `tibasMetaspace_ListenSpaceMemberEvents()`

tibasMetaspace_ListenMemberEvents()

Function

```
Declaration      tibas_status tibasMetaspace_ListenSpaceMemberEvents
                   (tibasMetaspace  metaspace,
                    tibasListener    listener)
```

Purpose	Attaches a listener to a metaspace for receiving metaspace member events.
----------------	---

Parameters

Parameter	Description
<code>metaspace</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
<code>listener</code>	The listener.

Remarks	Use the <code>tibasMetaspace_ListenSpaceMemberEvents()</code> function to attach a listener to a specified metaspace to listen for member events. You do not have to be joined to the space to call the function.
----------------	---

Before you can attach a listener to a space, there must be a listener object created by a call to the `tibasListener_Create()` function.

Specify this object with the `listener` parameter.

See Also [tibasMetaspace_ListenSpaceMemberEvents\(\)](#)

tibasMetaspace_ListenSpaceRemoteMemberEvents()

Function

Declaration `tibas_status tibasMetaspace_ListenSpaceRemoteMemberEvents`
 `(tibasMetaspace metaspace,`
 `const char* spaceName,`
 `tibasListener listener)`

Purpose Attaches a listener to a metaspace for receiving remote space member events.

Parameters	Parameter	Description
	<code>metaspace</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>spaceName</code>	The name of the space on which to attach the listener.
	<code>listener</code>	The listener.

Remarks Use the `tibasMetaspace_ListenSpaceMemberRemoteEvents()` function to attach a listener to a specified metaspace to return remote space member events. You do not have to be joined to the space to call the function.

Before you can attach a listener to a space, there must be a listener object created by a call to the `tibasListener_Create()` function.

Specify this object with the `listener` parameter.

See Also `tibasMetaspace_ListenSpaceMemberEvents()`

tibasMetaspace_ListenRemoteMemberEvents()

Function

Declaration `tibas_status` `tibasMetaspace_ListenRemoteMemberEvents`
 (`tibasMetaspace` `metaspace`,
 `tibasListener` `listener`)

Purpose Attaches a listener to a metaspace for receiving remote member events.

Parameters	Parameter	Description
	<code>metaspace</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>listener</code>	The listener.

Remarks Use the `tibasMetaspace_ListenRemoteMemberEvents()` function to attach a listener to a specified metaspace to listen for remote member events. You do not have to be joined to the space to call the function.

Before you can attach a listener to a space, there must be a listener object created by a call to the `tibasListener_Create()` function.

Specify this object with the `listener` parameter.

See Also `tibasMetaspace_ListenSpaceMemberEvents()`

tibasMetaspace_ListenSpaceState()

Function

Declaration `tibas_status` `tibasMetaspace_ListenSpaceState`
 (`tibasMetaspace` `metaspace`,
 `const char*` `spaceName`,
 `tibasListener` `listener`)

Purpose Attaches a listener to a metaspace for receiving space state events.

Parameters	Parameter	Description
	<code>metaspace</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>spaceName</code>	The name of the space on which to attach the listener.
	<code>listener</code>	The listener.

Remarks Use the `tibasMetaspace_ListenSpaceState()` function to attach a listener to a specified metaspace to listen for space state events. You do not have to be joined to the space to call the function.

Before you can attach a listener to a space, there must be a listener object created by a call to the `tibasListener_Create()` function.

Specify this object with the `listener` parameter.

See Also `tibasMetaspace_ListenSpaceMemberEvents()`

tibasMetaspace_GetRemoteListen()

Function

Declaration `tibas_status` tibasMetaspace_GetRemoteListen(
 `tibasMetaspace` metaspace,
 `char**` remoteListen)

Purpose Returns the listen URL for a specified metaspace.

Parameters	Parameter	Description
	metaspace	The Metaspace object that identifies the metaspace for which to return the listen URL.
	remoteListen	Returns the remote listen URL for the specified metaspace.

Remarks Use the `tibasMetaspace_GetRemoteListen()` function to return the remote listen URL for a specified metaspace space member.

See Also [tibasMemberDef_SetRemoteListen\(\)](#)

tibasMetaspace_Browse()

Function

Declaration

```
tibas_status tibasMetaspace_Browse(  
    tibasMetaspace    metaspace,  
    tibasBrowser*     browser,  
    const char*       spaceName,  
    tibas_browserType browserType,  
    tibasBrowserDef   browserDef,  
    const char*       filter)
```

Purpose

Creates a space browser on the specified space.

Parameters

Parameter	Description
metaspace	Specifies a Metaspace object identifying the metaspace containing the space which is to be browsed.
browser	The returned browser object.
spaceName	The name of the space to browse.
browserType	The type of the browser. Can be: TIBAS_BROWSER_GET, TIBAS_BROWSER_TAKE, or TIBAS_BROWSER_LOCK.
browserDef	A browser definition object.
filter	A string containing a filter to apply to the browser or NULL.

Remarks

Use the `tibasMetaspace_Browse()` function to create a space browser on a specified space. You do not have to be joined to the space to call the function.

Before you can call `tibasMetaspace_Browse()`, you must create a `browserDef` object by calling the [tibasBrowserDef_Create\(\)](#) function or the [tibasEventBrowserDef_CreateEx\(\)](#) function. You can then call additional `tibasBrowserDef` functions to specify the distribution scope and time scope for the browser and also specify a timeout value for the browser.

The `browserType` parameter specifies the type of browser, which can be one of the following:

- **TIBAS_BROWSER_GET** Specifies a Get Browser, which retrieves the next tuple in a series of tuples.
- **TIBAS_BROWSER_TAKE** Specifies a Take Browser, which retrieves the next tuple in a series of tuples and consumes it.
- **TIBAS_BROWSER_LOCK** Specifies a Lock Browser, which retrieves the next tuple in a series of tuples and locks it.

The `filter` parameter specifies a filter string that controls what data is browsed.



Calling `tibasMetaspace_Browse()` may cause the space to be automatically joined with a distribution role of `TIBAS_DISTRIBUTION_ROLE_LEECH`, if the space specified with the `spaceName` parameter has not already been joined.

See Also

[tibasBrowserDef_Create\(\)](#), [tibasEventBrowserDef_CreateEx\(\)](#),
[tibasEventBrowserDef_SetDistributionScope\(\)](#),
[tibasEventBrowserDef_SetTimeScope\(\)](#), [tibasEventBrowserDef_SetTimeout\(\)](#),
[tibasFilter_Create\(\)](#)

tibasMetaspace_BrowseEvents()

Function

Declaration `tibas_status` tibasMetaspace_BrowseEvents(
 tibasMetaspace metaspace,
 tibasEventBrowser* eventBrowser,
 const char* spaceName,
 tibasEventBrowserDef eventBrowserDef,
 const char* filter)

Purpose Creates an event browser on the specified space. May cause the space to be automatically joined with a distribution role of TIBAS_DISTRIBUTION_ROLE_LEECH if not already joined.

Parameters	Parameter	Description
	metaspace	The TIBCO ActiveSpaces entity on which the function is invoked.
	browser	The returned event browser object.
	spaceName	The name of the space to browse.
	browserDef	A browser definition object.
	filter	A string containing a filter to apply to the event browser or NULL.

Remarks Use the `tibasMetaspace_BrowseEvents()` function to create an event browser on a specified space.

Before you call this function, create an `eventBrowserDef` object by calling the [tibasEventBrowserDef_Create\(\)](#) or [tibasEventBrowserDef_CreateEx\(\)](#) function and define the attributes of the event browser. You can do this directly when you specify the arguments for the [tibasEventBrowserDef_CreateEx\(\)](#) function, or by calling the `EventBrowserDef` functions, which let you specify the distribution scope and time scope plus a timeout value for the event browser.

You can use the `filter` parameter to specify a filter that limits the events returned by the event browser.

After you create the event browser, use the [tibasEventBrowser_Next\(\)](#) function to iteratively return the next matching event in the space's event stream.

See Also [tibasEventBrowserDef_Create\(\)](#), [tibasEventBrowserDef_CreateEx\(\)](#),
[tibasEventBrowserDef_SetDistributionScope\(\)](#),
[tibasEventBrowserDef_GetTimeScope\(\)](#), [tibasEventBrowserDef_SetTimeout\(\)](#),
[tibasEventBrowser_Next\(\)](#)

tibasMetaspace_BeginTransaction()

Function

Declaration `tibas_status tibasMetaspace_BeginTransaction
(tibasMetaspace metaspace)`

Purpose Starts a transaction for the current thread.

Parameters

Parameter	Description
metaspace	The TIBCO ActiveSpaces entity on which the function is invoked.

Remarks Use the `tibasMetaspace_BeginTransaction()` function to begin a transaction for the current thread initiated by your application. Transactions are useful in the ActiveSpaces environment because space operations typically involve multiple tuples or affect multiple spaces.

After you begin the transaction, ActiveSpaces queues all space operations in the current thread in the transaction until you call either the `tibasMetaspace_CommitTransaction()` function to commit the transaction or the `tibasMetaspace_RollbackTransaction()` function to roll back the transaction. At that point, the transaction is ended.

If none of the transactions failed, your application should call `tibasMetaspace_CommitTransaction()`. If one or more operations failed, call `tibasMetaspace_RollbackTransaction()` to roll back the transaction.

Before you call `tibasMetaSpace_BeginTransaction()`, start a thread using a C language `BeginThread` function or a similar method.

You can also release the operations that have been added to the transaction for later acquisition by another application thread by calling the `tibasMetaspace_ReleaseContext()` function and then calling the `tibasMetaspace_AcquireContext()` function transfer the operations to a another thread.

See Also `tibasMetaspace_CommitTransaction()`, `tibasMetaspace_RollbackTransaction()`, `tibasMetaspace_ReleaseContext()`, `tibasMetaspace_AcquireContext()`

tibasMetaspace_CommitTransaction()

Function

```
Declaration      tibas_status tibasMetaspace_CommitTransaction
                   (tibasMetaspace metaspace)
```

Purpose Commits all the space operations invoked since the `tibasMetaspace_BeginTransaction()` function or the `tibasMetaspace_AcquireContext()` function was called.

Parameters

Parameter	Description
<code>metaspace</code>	The TIBCO ActiveSpaces entity on which the function is invoked.

Remarks	<p>Use the <code>tibasMetaspace_CommitTransaction()</code> function to commit a transaction that has been initiated by the <code>tibasMetaspace_BeginTransaction()</code> function. When you commit the transaction, all of the operations accumulated in the current thread where <code>tibasMetaspace_BeginTransaction()</code> was invoked are executed.</p> <p>In general, call <code>tibasMetaspace_CommitTransaction()</code> if none of the operations in the transaction failed; otherwise, call <code>tibasMetaspaceRollbackTransaction()</code> to roll back the operations.</p>
----------------	--

See Also [tibasMetaspace_BeginTransaction\(\)](#), [tibasMetaspace_RollbackTransaction\(\)](#), [tibasMetaspace_ReleaseContext\(\)](#), [tibasMetaspace_AcquireContext\(\)](#)

tibasMetaspace_RollbackTransaction()

Function

Declaration `tibas_status` tibasMetaspace_RollbackTransaction
(tibasMetaspace metaspace)

Purpose Rolls back all of the space operations invoked since the
`tibasMetaspace_BeginTransaction()` function or the
`tibasMetaspace_AcquireContext()` function was called.

Parameters	Parameter	Description
	metaspace	Specifies the metaspace in which the transaction was started.

Remarks Use the `tibasMetaspace_RollbackTransaction()` function to roll back the space operations that were accumulated in a transaction since the `tibasMetaspace_BeginTransaction()` function or the `tibasMetaspace_AcquireContext()` function) was called.

You can use this function to roll back the operations that you committed, in the event that any of the transactions fail. After the transactions are rolled back, any changes made to the data grid are discarded.

See Also `tibasMetaspace_BeginTransaction()`, `tibasMetaspace_CommitTransaction()`,
`tibasMetaspace_ReleaseContext()`, `tibasMetaspace_AcquireContext()`

tibasMetaspace_ReleaseContext()

Function

Declaration `tibas_status tibasMetaspace_ReleaseContext
 (tibasMetaspace metaspace,
 tibasContext* context)`

Purpose Releases the thread context from a specified thread that called `tibasMetaspace_BeginTransaction()`.

Parameters

Parameter	Description
metaspace	The TIBCO ActiveSpaces entity on which the function is invoked.
context	Specifies the thread context.

Remarks Use the `tibasMetaspace_ReleaseContext()` function to release thread context for a specified current transaction for an application thread.

 You can then call the `tibasMetaspace_AcquireContext()` to acquire the thread context for another thread.

See Also `tibasMetaspace_BeginTransaction()`, `tibasMetaspace_CommitTransaction()`, `tibasMetaspace_RollbackTransaction()`, `tibasMetaspace_AcquireContext()`

tibasMetaspace_AcquireContext()

Function

Declaration `tibas_status tibasMetaspace_AcquireContext
(tibasMetaspace metaspace,
tibasContext* context)`

Purpose Acquires a thread context that has been released by another thread through a call to the [tibasMetaspace_ReleaseContext\(\)](#) function.

Parameters	Parameter	Description
	metaspace	The TIBCO ActiveSpaces entity on which the function is invoked.
	context	The thread context to acquire.

Remarks Use the `tibasMetaspace_AcquireContext()` function to acquire a thread context that has been released by another thread through a call to the [tibasMetaspace_ReleaseContext\(\)](#) function.

The `ReleaseContext` and `AcquireContext` functions allow your application to transfer thread contexts, including access to locks and to the current transaction.

See Also [tibasMetaspace_BeginTransaction\(\)](#), [tibasMetaspace_CommitTransaction\(\)](#), [tibasMetaspace_ReleaseContext\(\)](#), [tibasMetaspace_RollbackTransaction\(\)](#)

tibasMetaspace_GetRemoteMembers()

Function

Declaration `tibas_status tibasMetaspace_GetRemoteMembers(
 tibasMetaspace metaspace,
 tibasMemberList* memberList)`

Purpose Returns a list of the remote members of the metaspace.

Parameters	Parameter	Description
	metaspace	Specifies a Metaspace object that points to the metaspace for which to return a list of remote members.
	memberList	A returned list of remote members of the metaspace.

Remarks Use the `tibasMetaspace_GetRemoteMembers()` function to return a list of the remote members of a specified metaspace.

See Also [tibasMetaspace_GetMembers\(\)](#)

tibasMetaspace_GetMembers()

Function

Declaration

```
tibas_status tibasMetaspace_GetMembers(  
    tibasMetaspace metaspace,  
    tibasMemberList* memberList)
```

Purpose

Returns a list of the members of the metaspace.

Parameters

Parameter	Description
metaspace	The TIBCO ActiveSpaces entity on which the function is invoked.
memberList	A returned list of members of the metaspace.

Remarks

Use the `tibasMetaspace_GetMembers()` function to return a list of the members of a specified metaspace.

See Also

[tibasMetaspace_GetRemoteMembers\(\)](#)

tibasMetaspace_GetSpaceMembers()

Function

Declaration

```
tibas_status tibasMetaspace_GetSpaceMembers(  
    tibasMetaspace    metaspace,  
    tibasMemberList*  memberList,  
    const char*        spaceName)
```

Purpose

Returns a list of all the members of the space.

Parameters

Parameter	Description
metaspace	The TIBCO ActiveSpaces entity on which the function is invoked.
memberList	A returned list of members of the metaspace.
spaceName	The name of the space.

Remarks

Use the `tibasMetaspace_GetSpaceMembers()` function to return a list of the members of a specified space in a specified metaspace.

After you have obtained the member list, you can call the `tibasMemberList` functions to determine additional information about the members.

See Also

[tibasMetaspace_GetSystemSpace\(\)](#), [tibasMetaspace_GetSpaceDef\(\)](#), [tibasMemberList_Size\(\)](#), [tibasMemberList_Get\(\)](#)

tibasMetaspace_GetSelfMember()

Function

```
Declaration      tibas_status tibasMetaspace_GetSelfMember
                   (tibasMetaspace  metaspace,
                    tibasMember*     member)
```

Purpose	Returns the member for the current metaspace connection.
----------------	--

Parameters

Parameter	Description
metaspace	The TIBCO ActiveSpaces entity on which the function is invoked.
member	The member returned by the function.

Remarks	Use the <code>tibasMetaspace_GetSelfMember()</code> function to return the name of the space member currently connected to a specified metaspace.
----------------	---

After you have obtained the member object, you can use it as input to the `tibasMember` functions, to obtain additional information about the member; for example, you can call `tibasMember_GetManagementRole()` to determine the management role for the member and call `tibasMember_GetDistributionRole()` to return the distribution role.

See Also [tibasMember_Free\(\)](#), [tibasMember_GetName\(\)](#), [tibasMember_GetManagementRole\(\)](#), [tibasMember_Free\(\)](#), [tibasMember_GetDistributionRole\(\)](#)

tibasMetaspace_GetLogLevel()

Function

Declaration `tibas_status` `tibasMetaspace_GetLogLevel`
 (`tibasMetaspace` `metaspace`,
 `tibas_loglevel` `level`)

Purpose Returns the log level set for the metaspace.

Parameters	Parameter	Description
	<code>metaspace</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>level</code>	The log level returned by the function. The possible values are <code>TIBAS_LOG_FATAL</code> , <code>TIBAS_LOG_FINE</code> , <code>TIBAS_LOG_FINER</code> , <code>TIBAS_LOG_FINEST</code> , <code>TIBAS_LOG_ERROR</code> , <code>TIBAS_LOG_WARN</code> , or <code>TIBAS_LOG_INFO</code> .

Remarks Use the `tibasMetaspace_GetLogLevel()` function to determine the log level that is currently set for a specified metaspace. After determining the log level, you can change it if needed by calling the `tibasMetaspace_SetLogLevel()` function.

See Also `tibasMetaspace_SetLogLevel()`

tibasMetaspace_GetDiscovery()

Function

Declaration `tibas_status tibasMetaspace_GetDiscovery
(tibasMetaspace metaspace,
char** discovery)`

Purpose Returns the discovery URL for the metaspace connection.

Parameters

Parameter	Description
metaspace	Specifies the metaspace for which you want to return the discovery address.
discovery	Returns a string in one of the supported formats for discovery.

Remarks Use the `tibasMetaspace_GetDiscovery()` function to return the discovery address that is configured for a specified metaspace.

TIBCO ActiveSpaces uses three discovery protocols. Each protocol requires a different discovery URL format:

- TCP Discovery URL format
- PGM (Pragmatic General Multicast) URL format
- TIBCO Rendezvous Discovery URL format

For detailed information on the discovery formats, see [Discovery Attribute](#) on [page 51](#) of the *TIBCO ActiveSpaces Developer's Guide*.

The program must not modify or free the string.

See Also [tibasMemberDef_GetDiscovery\(\)](#), [tibasMemberDef_SetDiscovery\(\)](#)

tibasMetaspace_GetListen()

Function

Declaration	<code>tibas_status</code> tibasMetaspace_GetListen(tibasMetaspace metaspace, char** listen)						
Purpose	Returns the listen attribute defined for a specified metaspace connection.						
Parameters	<table><tr><th>Parameter</th><th>Description</th></tr><tr><td>metaspace</td><td>The TIBCO ActiveSpaces entity on which the function is invoked.</td></tr><tr><td>listen</td><td>A string in the format tcp://interface/port The program must not modify or free the string.</td></tr></table>	Parameter	Description	metaspace	The TIBCO ActiveSpaces entity on which the function is invoked.	listen	A string in the format tcp://interface/port The program must not modify or free the string.
Parameter	Description						
metaspace	The TIBCO ActiveSpaces entity on which the function is invoked.						
listen	A string in the format tcp://interface/port The program must not modify or free the string.						
Remarks	<p>Use the <code>tibasMetaspace_GetDiscovery()</code> function to return the listen attribute currently active for a specified metaspace.</p> <p>The listen attribute is specified in the MemberDef object that defines the characteristics of the metaspace. You can use the <code>tibasMemberDef_SetListen()</code> function to set the listen attribute and the <code>tibasMetaspace_GetListen()</code> function to return the listen attribute currently set in the MemberDef object.</p>						
See Also	<code>tibasMetaspace_DefineSpace()</code> , <code>tibasMemberDef_GetListen()</code> , <code>tibasMemberDef_SetListen()</code>						

tibasMetaspace_GetRemoteDiscovery()

Function

Declaration

```
tibas_status tibasMetaspace_GetRemoteDiscovery(  
    tibasMetaspace    metaspace,  
    char**             remoteDiscovery)
```

Purpose

Gets the remote discovery URL for a specified metaspace.

Parameters

Parameter	Description
metaspace	Specify the name of the metaspace for which to get the remote discovery URL.
remoteDiscovery	Returns the remote discovery URL.

Remarks

Use the `tibasMetaspace_GetRemoteDiscovery()` function to return the remote discovery URL that is set up for a specified metaspace.

See Also

[tibasMemberDef_SetRemoteDiscovery\(\)](#),
[tibasMemberDef_GetRemoteDiscovery\(\)](#)

tibasMetaspace_SetLogLevel()

Function

Declaration `tibas_status tibasMetaspace_SetLogLevel`
 (`tibasMetaspace metaspace,`
 `tibas_logLevel level`)

Purpose Sets the log level for the metaspace.

Parameters	Parameter	Description
	<code>metaspace</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>level</code>	The log level passed to the function. The possible values are TIBAS_LOG_FATAL, TIBAS_LOG_ERROR, TIBAS_LOG_FINE, TIBAS_LOG_FINER, TIBAS_LOG_FINEST, TIBAS_LOG_WARN, or TIBAS_LOG_INFO.

Remarks Use the `tibasMetaspace_SetLogLevel()` function to set the log level for a specified metaspace.

See Also [tibasMetaspace_GetLogLevel\(\)](#)

tibasMetaspace_Free()

Function

Declaration `tibas_status tibasMetaspace_Free
(tibasMetaspace* metaspace)`

Purpose Frees resources used by the Metaspace object.

Parameter

Parameter	Description
metaspace	The TIBCO ActiveSpaces entity on which the function is invoked.

Remarks Use the `tibasMetaspace_Free()` function to free a specified Metaspace object. Calling this function immediately closes the connection to the metaspace if all spaces, browsers, and listeners have been freed, or when they are all freed.

See Also [tibasMetaspace_Connect\(\)](#)

tibas_FreeData()

Function

Declaration `tibas_status` `tibas_FreeData`
 (`char** data`)

Purpose Frees resources given to the user.

Parameters	Parameter	Description
	<code>data</code>	A pointer to data allocated by ActiveSpaces to free. Usually a string, blob, or array.

Remarks Use the `tibas_FreeData()` function to free data that has been allocated to your application.

Chapter 3 **Space**

A space provides shared virtual storage for data.

- A space is a shared entity in the sense that many applications can access it concurrently and each application has the same coherent view of the data contained in the space. The spaces in ActiveSpaces are called tuple spaces, and the items stored in them are called tuples.
- A space is a virtual entity in the sense that it is distributed and implemented collaboratively by a group of processes located on multiple hosts and communicating over the network.

The space operations that perform data operations on the space generally are provided as pairs of functions—a base version of the operation and an “extended” version of the operation that provides more flexibility in performing the operation.

Topics

- [Space Operations and Options Initialization Operations](#)

Space Operations and Options Initialization Operations

The following table lists the space operations:

Table 7 Space

Function or Type	Description	Page
<code>tibasSpace_Browse()</code>	Returns a browser for the space to iterate through all the tuples that match the filter.	70
<code>tibasSpace_BrowseEvents()</code>	Returns a browser for the space to iterate through all the events that match the filter.	72
<code>tibasSpace_CompareAndPut()</code>	Specifies a tuple to look for. If the old value is unmodified, replaces the existing tuple with the specified value.	73
<code>tibasSpace_CompareAndPutEx()</code>	Compares the values in a specified tuple with existing data. If the old value is unmodified, replaces the existing tuple with the specified value. Allows you to set lock and lock wait attributes for the tuple.	74
<code>tibasSpace_CompareAndPutAll()</code>	Compares the values in a list of old values with the existing values in the space. If the old values are unmodified, replaces the existing tuples with the specified new values.	76
<code>tibasSpace_CompareAndPutAllEx()</code>	Compares the values in a list of old values with the existing values in the space. If the old values are unmodified, replaces the existing tuples with the specified new values. Allows you to set lock and lock wait attributes for the tuple.	77
<code>tibasSpace_CompareAndTake()</code>	Looks for a tuple containing a specified value, and if it is found, takes the tuple value.	79
<code>tibasSpace_CompareAndTakeEx()</code>	Looks for a tuple containing a specified value, and if it is found, takes the tuple value. Allows you to set lock and lock wait attributes for the tuples.	80

Table 7 *Space*

Function or Type	Description	Page
<code>tibasSpace_CompareAndTakeAll()</code>	Compares values in a list of specified values with the values in the existing tuples. If value matches are found, takes the specified tuples.	82
<code>tibasSpace_CompareAndTakeAllEx()</code>	Compares values in a list of specified values with the values in the existing tuples. If value matches are found, takes the specified tuples. Allows you to set lock and lock wait attributes for the tuples.	83
<code>tibasSpace_Get()</code>	Returns the tuple (if one exists) in the specified space whose key fields match the key fields of a specified tuple.	66
<code>tibasSpace_GetAll()</code>	Batch version of the <code>tibasSpace_Get()</code> function. Returns a list of tuples matching key values that you provide.	69
<code>tibasSpace_GetName()</code>	Returns the space name.	99
<code>tibasSpace_GetMetaspaceName()</code>	Returns the metaspace name for the metaspace to which a specified space belongs..	100
<code>tibasSpace_GetMetaspace()</code>	Returns the metaspace to which a specified space belongs.	101
<code>tibasSpace_GetSpaceDef()</code>	Returns the <code>spaceDef</code> object that defines a specified space.	98
<code>tibasSpace_IsReady()</code>	Sets the value of <code>isReady</code> to <code>TIBAS_TRUE</code> if the space is ready to be used.	67
<code>tibasSpace_Listen()</code>	Attaches a listener to a space for receiving space events.	85
<code>tibasSpace_Load()</code>	Loads a tuple into a specified space, but does not trigger a persister's <code>onWrite</code> function. Used in the <code>Onload</code> function provided to implement shared all persistence.	116

Table 7 Space

Function or Type	Description	Page
<code>tibasSpace_LoadAll()</code>	Batch version of the <code>tibasSpace_Load()</code> function. Used if shared all persistence is implemented.	117
<code>tibasSpace_Lock()</code>	Locks the tuple associated with a specified key value.	103
<code>tibasSpace_LockEx()</code>	Locks the tuple associated with a key tuple value, and allows you to specify a wait value and a forget value.	104
<code>tibasSpace_LockAll()</code>	Batch version of the Lock operation. Requires a list of key value tuples to lock and returns a list of the tuples that are locked.	105
<code>tibasSpace_LockAllEx()</code>	Batch version of the lock operation. Requires a list of key value tuples to take and returns a list of results.	106
<code>tibasSpace_Unlock()</code>	Unlocks a tuple in a specified space that has a specified key.	107
<code>tibasSpace_UnlockAll()</code>	Batch version of the unlock operation. Requires a list of tuples to unlock and returns a list of results.	108
<code>tibasSpace_Put()</code>	Stores a tuple into the space and returns the corresponding tuple.	86
<code>tibasSpace_PutEx()</code>	Stores a tuple into the space and returns the corresponding tuple, and allows you to specify a lock wait value, lock or unlock value, and a forget value.	87
<code>tibasSpace_PutAll()</code>	The batch version of the put operation. Stores all of the tuples in the provided list and returns a list of results.	89

Table 7 *Space*

Function or Type	Description	Page
<code>tibasSpace_PutAllEx()</code>	The enhanced batch version of the put operation. Stores all of the tuples in the provided list and returns a list of results. Allows you to specify a lock wait value, a lock or unlock value, and a forget value.	90
<code>tibasSpace_Take()</code>	Returns and atomically removes from the space the tuple (if one exists) whose tuple key fields match the key fields of the tuple provided.	92
<code>tibasSpace_TakeEx()</code>	Returns and atomically removes from the space the tuple (if one exists) whose tuple key fields match the key fields of the tuple provided. Allows you to specify a lock wait value, a lock and unlock value, and a forget value.	93
<code>tibasSpace_TakeAll()</code>	Batch version of the take operation,. Requires a list of key value tuples to take and returns a list of results.	95
<code>tibasSpace_TakeAllEx()</code>	Batch version of the take operation. Requires a list of key value tuples to take and returns a list of results. Allows you to specify a lock wait value, a lock or unlock value, and a forget value.	96
<code>tibasSpace_SetDistributionRole()</code>	Sets the distribution role of the member for the space.	109
<code>tibasSpace_SetPersister()</code>	Register the persister object being specified as a persister on the space.	110
<code>tibasSpace_Invoke()</code>	Invokes an appropriate function defined in the application code of a member that seeds a specified space.	115
<code>tibasSpace_InvokeMember()</code>	Invokes a specified application function on the members of a specified space.	113

Table 7 *Space*

Function or Type	Description	Page
tibasSpace_InvokeMembers()	Invokes a specified application function on all seeders that have joined a specified space.	115
tibasSpace_InvokeRemoteMembers()	This is a deprecated function.	114
tibasSpace_InvokeSeeders()	Invokes a specified application function on all seeders that have joined a specified space.	115
tibasSpace_Size()	Returns the number of entries in the space.	102
tibasSpace_WaitForReady()	Queries the state of the space. Returns a value indicating that the space is ready; or if it is not ready, returns a value indicating that the space is not ready.	68

The following table lists the options initialization operations:

Table 8 *Options Initialization*

Function or Type	Description	Page
tibasRecoveryOptions_Initialize()	Sets the values in a structure used to call the tibasMetaspace_RecoverSpaceEx() function.	119
tibasPutOptions_Initialize()	Sets the values in a structure used to call the tibasSpace_PutAllEx() function or the tibasSpace_CompareAndPutAllEx() function.	120
tibasTakeOptions_Initialize()	Sets the values in a structure used to call the tibasSpace_TakeAllEx() function or the tibasSpace_CompareAndTakeAllEx() function.	121
tibasLockOptions_Initialize()	Sets the values in a structure used to call the tibasSpace_LockEx() function.	122
tibasUnlockOptions_Initialize()	Sets the values in a tibasUnlockOptions structure.	123
tibasInvokeOptions_Initialize()	Sets the values in a structure used to call the tibasSpace_Invoke() function or the tibasSpace_InvokeMember() function.	124

tibasSpace_Get()

Function

Declaration `tibas_status` `tibasSpace_Get`
 (`tibasSpace` `space`,
 `tibasTuple*` `value`,
 `tibasTuple` `key`)

Purpose Returns the tuple (if one exists) in the specified space whose key fields match the key fields of a specified tuple.

Parameters	Parameter	Description
	<code>space</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>value</code>	Pointer to the tuple returned by the function.
	<code>key</code>	A tuple containing one or more key fields.

Remarks Use the `tibasSpace_Get()` function to get a tuple whose key matches the key in a specified tuple.

See Also `tibasSpace_GetAll()`, `tibasSpace_CompareAndPut()`,
 `tibasSpace_CompareAndPutEx()`,
 `tibasSpace_CompareAndPutAll()`,`tibasSpace_CompareAndPutAllEx()`,
 `tibasSpace_CompareAndTake()`, `tibasSpace_CompareAndTakeEx()`,
 `tibasSpace_CompareAndTakeAll()`, `tibasSpace_CompareAndTakeAllEx()`,
 `tibasSpace_Put()`, `tibasSpace_PutEx()`, `tibasSpace_PutAll()`.
 `tibasSpace_CompareAndPutAllEx()`, `tibasSpace_Take()`, `tibasSpace_TakeEx()`,
 `tibasSpace_TakeAll()`, `tibasSpace_TakeAllEx()`

tibasSpace_IsReady()

Function

Declaration `tibas_status` `tibasSpace_IsReady`
 (`tibasSpace` `space`,
 `tibas_boolean` `*isReady`)

Purpose Sets the value of `isReady` to `TIBAS_TRUE` if the space is ready to be used.

Parameters	Parameter	Description
	<code>space</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>isReady</code>	Boolean value indicating whether the space is ready to be used.

Remarks Use the `tibasSpace_IsReady()` function to test whether a specified space is ready to be used.

A space is ready to be used if the minimum number of seeders has been reached or, if persistence has been configured for the space, the space has been loaded from the persistence layer.

If the space is not ready, the function returns `TIBAS_FALSE`, and it is not possible to perform operations on the space.

See Also [tibasSpace_WaitForReady\(\)](#)

tibasSpace_WaitForReady()

Function

Declaration `tibas_status` `tibasSpace_WaitForReady`
 (`tibasSpace` `space`,
 `tibas_boolean*` `isReady`,
 `tibas_long` `timeout`)

Purpose Queries the state of the space. Returns a value indicating that the space is ready; or if it is not ready, returns a value indicating that the space is not ready.

Parameters	Parameter	Description
	<code>space</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>isReady</code>	Is equal to <code>TIBAS_TRUE</code> if the space is ready, or <code>TIBAS_FALSE</code> if the space is not ready.
	<code>timeout</code>	Specify a value indicating the number of milliseconds to wait for the space to reach the ready state if the space is not ready.

Remarks Use the `tibasSpace_WaitForReady()` function to query whether a specified space is ready to process data. If the space is ready, the function returns `TIBAS_TRUE`. If the space is not ready, it returns `TIBAS_FALSE`

You can use the `timeout` parameter to specify a value indicating the number of milliseconds to wait before the space is ready.

See Also [tibasSpace_IsReady\(\)](#)

tibasSpace_GetAll()

Function

Declaration `tibas_status tibasSpace_GetAll
 (tibasSpace space,
 tibasSpaceResultList *resultList,
 tibasTupleList keyList)`

Purpose Batch version of the `tibasSpace_Get()` function. Returns a list of tuples matching key values that you provide.

Parameters

Parameter	Description
<code>space</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
<code>resultList</code>	The list of tuples returned by the function.
<code>keyList</code>	Provided list of tuples containing key fields.

Remarks Use the `tibasSpace_GetAll()` function to return a list of tuples that match key values that you provide.

To use `tibasSpace_GetAll()`, you must create and populate a `keyList` containing the keys to search for, which you then pass to the function in the `keyList` parameter. To create a `keyList`, you call the [tibasTupleList_Create\(\)](#) function. To add keys to the list, you call the [tibasTupleList_Put\(\)](#) function.

You must also initialize a `tibasSpaceResultList` object, which you pass to the function in the `resultList` parameter.

The function returns a `resultList` that contains the tuples returned by the get operations.

For more information on Result lists, see the reference articles in [Chapter 20, SpaceResult and InvokeResult on page 391](#) and in [Chapter 21, SpaceResultList and InvokeResultList on page 405](#).

After you are done with the `keyList`, free it by calling the [tibasTupleList_Free\(\)](#) function, and after you are done with the Result list, free it by calling the [tibasSpaceResultList_Free\(\)](#) function.

See Also [tibasSpace_Get\(\)](#), [tibasTupleList_Create\(\)](#), [tibasTupleList_Put\(\)](#), [tibasTupleList_Free\(\)](#), [tibasSpaceResultList_Free\(\)](#)

tibasSpace_Browse()

Function

Declaration

```
tibas_status tibasSpace_Browse(  
    tibasSpace space,  
    tibasBrowser* browser,  
    tibas_browserType browserType,  
    tibasBrowserDef browserDef,  
    const char* filter)
```

Purpose

Returns a browser for the space to iterate through all the tuples that match the filter.

On each iteration, the browser will apply an operation on the next tuple. The operation is either a get, take, or lock depending on the browser type.

Parameters

Parameter	Description
space	The TIBCO ActiveSpaces entity on which the function is invoked.
browser	Returns a browser object.
browserType	Can be TIBAS_BROWSER_GET, TIBAS_BROWSER_TAKE, or TIBAS_BROWSER_LOCK.
browserDef	The browser definition can be used to set timeout values and distribution or time scopes
filter	The filter criteria.

Remarks

Use the `tibasSpace_Browse()` function to create an event browser on a specified space.

The `browserType` parameter specifies the type of browser, which can be one of the following:

- **TIBAS_BROWSER_GET** Specifies a Get Browser, which retrieves the next tuple in a series of tuples.
- **TIBAS_BROWSER_TAKE** Specifies a Take Browser, which retrieves the next tuple in a series of tuples and consumes it.
- **TIBAS_BROWSER_LOCK** Specifies a Lock Browser, which retrieves the next tuple in a series of tuples and locks it.

Before you call this function, create a BrowserDef object by calling the [tibasBrowserDef_Create\(\)](#) or [tibasBrowserDef_CreateEx\(\)](#) function and define the attributes of the event browser. You can do this directly when you specify the arguments for the [tibasBrowserDef_CreateEx\(\)](#) function, or by calling the BrowserDef functions, which let you specify the distribution scope and time scope plus a timeout value for the browser.

You can use the `filter` parameter to specify a filter that limits the events returned by the event browser.

After you create the event browser, use the [tibasEventBrowser_Next\(\)](#) to iteratively return the next matching event in the space's event stream.

tibasSpace_BrowseEvents()

Function

Declaration `tibas_status` `tibasSpace_BrowseEvents`
 (`tibasSpace` `space`,
 `tibasEventBrowser*` `eventBrowser`,
 `tibasEventBrowserDef` `browserDef`,
 `const char*` `filter`)

Purpose Returns a browser for the space to iterate through all the events that match the filter.

Parameters	Parameter	Description
	<code>space</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>eventBrowser</code>	The browser returned by the function.
	<code>browserDef</code>	The browser definition can be used to set timeout values and distribution or time scopes
	<code>filter</code>	The filter criteria.

Remarks Use the `tibasSpace_BrowseEvents()` function to create an event browser on a specified space.

On each iteration, the browser will return an event for the next tuple. Events represent a put, take, seed, unseed, or expire event

Before you call this function, create an `eventBrowserDef` object by calling the `tibasEventBrowserDef_Create()` or `tibasEventBrowserDef_CreateEx()` function and define the attributes of the event browser. You can do this directly when you specify the arguments for the `tibasEventBrowserDef_CreateEx()` function, or by calling the `EventBrowserDef` functions, which let you specify the distribution scope and time scope plus a timeout value for the event browser.

You can use the filter parameter to specify a filter that limits the events returned by the event browser.

After you create the event browser, use the `tibasEventBrowser_Next()` function to iteratively return the next matching event in the space's event stream.

See Also `tibasEventBrowserDef_Create()`, `tibasEventBrowserDef_CreateEx()`,
 `tibasEventBrowserDef_SetDistributionScope()`,
 `tibasEventBrowserDef_GetTimeScope()`, `tibasEventBrowserDef_SetTimeout()`,
 `tibasEventBrowser_Next()`, `tibasMetaspace_BrowseEvents()`

tibasSpace_CompareAndPut()

Function

Declaration `tibas_status` `tibasSpace_CompareAndPut`(
 `tibasSpace` `space`,
 `tibasTuple*` `value`,
 `tibasTuple` `oldValue`,
 `tibasTuple` `newValue`)

Purpose Specifies a tuple to look for. If the old value is unmodified, replaces the existing tuple with the specified value.

Parameters	Parameter	Description
	<code>space</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>value</code>	Returns the value of the tuple.
	<code>oldValue</code>	A tuple containing the value to look for.
	<code>newValue</code>	A tuple containing a new value to replace the old value.

Remarks Use the `tibasSpace_CompareAndPut()` function to search for a specified tuple in a specified space and, if the value specified in the `oldValue` parameter is unmodified, replace the tuple with a specified new value.

If there is a mismatch, the function returns the existing value in the space. If the operation is successful, the returned tuple will be same as the new tuple passed to the function. Otherwise, the return will be a different object.

See Also [tibasSpace_CompareAndPutEx\(\)](#), [tibasSpace_CompareAndPutAll\(\)](#), [tibasSpace_CompareAndPutAllEx\(\)](#), [tibasSpace_Put\(\)](#), [tibasSpace_PutEx\(\)](#), [tibasSpace_CompareAndPutAllEx\(\)](#)

tibasSpace_CompareAndPutEx()

Function

Declaration

```
tibas_status tibasSpace_CompareAndPutEx(  
    tibasSpace space,  
    tibasTuple* value,  
    tibasTuple oldValue,  
    tibasTuple newValue,  
    tibasPutOptions options)
```

Purpose

Compares the values in a specified tuple with existing data. If the old value is unmodified, replaces the existing tuple with the specified value. Allows you to set lock and lock wait attributes for the tuple.

Parameters

Parameter	Description
space	The TIBCO ActiveSpaces entity on which the function is invoked.
value	Returns the value of the tuple that is located
oldValue	Specifies the value to look for. If you code NULL for this argument, an empty tuple is replaced.
newValue	Specifies the new value.
options	Provide a tibasPutOptions structure that specifies the options for the put.

Remarks

Use the `tibasSpace_CompareAndPutEx()` function to compare the value in a specified tuple with the value in the data store, and if the value is unmodified, replace the existing tuple with the specified value.

`tibasSpace_CompareAndPutEx()` also allows you to specify additional options for the operation, by passing a `tibasPutOptions` structure in the `options` parameter. The `tibasPutOptions` structure is defined as follows:

```
struct _tibasPutOptions {  
    tibas_long    entryTTL;  
    tibas_long    lockWait;  
    tibas_boolean lock;  
    tibas_boolean unlock;  
    tibas_boolean forget;  
};
```

- The put options are defined as follows:
- entryTTL** Allows you to define the entry TTL for the tuple that is being put.

- **lockWait** Specifies a lock wait time, in milliseconds.
- **lock** Put the data and lock the tuple.
- **unlock** Unlock the tuple. Only the thread or process that locked the tuple can unlock it.

Lock and unlock are mutually exclusive. If you use one of these, code NULL for the other.

- **forget** Specify this option if you want the system to forget the return. This is useful if you do not need the tuple.

See Also [tibasSpace_CompareAndPut\(\)](#), [tibasSpace_CompareAndPutAll\(\)](#), [tibasSpace_CompareAndPutAllEx\(\)](#)

tibasSpace_CompareAndPutAll()

Function

Declaration

```
tibas_status tibasSpace_CompareAndPutAll(  
    tibasSpace          space,  
    tibasSpaceResultList* resultList,  
    tibasTupleList      oldValueList,  
    tibasTupleList      newValueList)
```

Purpose

Compares the values in a list of old values with the existing values in the space. If the old values are unmodified, replaces the existing tuples with the specified new values.

Parameters

Parameter	Description
space	The TIBCO ActiveSpaces entity on which the function is invoked.
resultList	A list of the values in the space.
oldValueList	A list of old values.
newValueList	A list of new values to compare with the values in the list.

Remarks

Use the `tibasSpace_CompareAndPutAll()` function to compare values that you specify in a list of new values with the existing values for the tuples in the data store, and, if the new values do not match an existing values, replace the old values with the new values.

See Also

[tibasSpace_CompareAndPutEx\(\)](#), [tibasSpace_CompareAndPutAll\(\)](#), [tibasSpace_CompareAndPutAllEx\(\)](#)

tibasSpace_CompareAndPutAllEx()

Function

Declaration

```
tibas_status tibasSpace_CompareAndPutAllEx(
    tibasSpace          space,
    tibasSpaceResultList* resultList,
    tibasTupleList      oldValueList,
    tibasTupleList      newValueList,
    tibasPutOptions      options)
```

Purpose Compares the values in a list of old values with the existing values in the space. If the old values are unmodified, replaces the existing tuples with the specified new values. Allows you to set lock and lock wait attributes for the tuple.

Parameters

Parameter	Description
space	The TIBCO ActiveSpaces entity on which the function is invoked.
resultList	A list of the values in the space.
oldValueList	A list of old values. If you specify NULL in a list item, only empty tuples are replaced.
newValueList	A list of new values to compare with the values in the list.
options	Provide a tibasPutOptions structure that specifies the options for the put.

Remarks

Use the `tibasSpace_CompareAndPutAllEx()` function to compare the values in the tuples in a space with values stored in an old value list, and if a value is unmodified, replace it with the value in the `newValueList`. Allows you to set lock and lock wait attributes for the tuples.

`tibasSpace_CompareAndPutAllEx()` also allows you to specify additional options for the operation, by passing a `tibasPutOptions` structure in the `options` parameter. The `tibasPutOptions` structure is defined as follows:

```
struct _tibasPutOptions {
    tibas_long    entryTTL;
    tibas_long    lockWait;
    tibas_boolean lock;
    tibas_boolean unlock;
    tibas_boolean forget;
};
```

The put options are defined as follows:

- **entryTTL** Allows you to define the entry TTL for the tuple that is being put.

- **lockWait** Specifies a lock wait time, in milliseconds.
- **lock** Put the data and lock the tuple.
- **unlock** Unlock the tuple. Only the thread or process that locked the tuple can unlock it.

Lock and unlock are mutually exclusive. If you use one of these, code NULL for the other.

- **forget** Specify this option if you want the system to forget the return. This is useful if you do not need the tuple.

See Also [tibasSpace_CompareAndPut\(\)](#), [tibasSpace_CompareAndPutEx\(\)](#), [tibasSpace_CompareAndPutAll\(\)](#)

tibasSpace_CompareAndTake()

Function

Declaration

```
tibas_status tibasSpace_CompareAndTake(  
    tibasSpace    space,  
    tibasTuple*   value,  
    tibasTuple    oldValue)
```

Purpose

Looks for a tuple containing a specified value, and if it is found, takes the tuple value.

Parameters

Parameter	Description
space	The TIBCO ActiveSpaces entity on which the function is invoked.
value	Returns the value that is taken.
oldValue	Specifies the value to compare for.

Remarks

Use the `tibasSpace_CompareAndTake()` function to look for a tuple containing a specified value, and if it is found, take the existing data.

See Also

[tibasSpace_CompareAndTakeEx\(\)](#), [tibasSpace_CompareAndTakeAll\(\)](#), [tibasSpace_CompareAndTakeAllEx\(\)](#)

tibasSpace_CompareAndTakeEx()

Function

Declaration `tibas_status` `tibasSpace_CompareAndTakeEx`(
 `tibasSpace` `space`,
 `tibasTuple*` `value`,
 `tibasTuple` `oldValue`,
 `tibasTakeOptions` `options`)

Purpose Looks for a tuple containing a specified value, and if it is found, takes the tuple value. Allows you to set lock and lock wait attributes for the tuples.

Parameters	Parameter	Description
	<code>space</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>value</code>	Returns the value that is taken.
	<code>oldValue</code>	Specifies the value to compare for.
	<code>options</code>	Provide a <code>tibasTakeOptions</code> structure that specifies the options for the take.

Remarks Use the `tibasSpace_CompareAndTakeEx()` function to look for a tuple containing a specified value, and if it is found, take the tuple with the specified value.

`tibasSpace_CompareAndTakeEx()` also allows you to specify additional options for the operation, by passing a `tibasTakeOptions` structure in the `options` parameter. The `tibasTakeOptions` structure is defined as follows:

```
struct _tibasTakeOptions {
    tibas_long    lockWait;
    tibas_boolean lock;
    tibas_boolean unlock;
    tibas_boolean forget;
};
```

The take options are defined as follows:

- **lockWait** Specifies a lock wait time, in milliseconds.
- **lock** Take the data and lock the tuple.
- **unlock** Unlock the tuple. Only the thread or process that locked the tuple can unlock it.

Lock and unlock are mutually exclusive. If you use one of these, code NULL for the other.

- **forget** Specify this option if you want the system to forget the return. This is useful if you do not need the tuple.

tibasSpace_CompareAndTakeAll()

Function

Declaration `tibas_status` `tibasSpace_CompareAndTakeAll`(
 `tibasSpace` `space`,
 `tibasSpaceResultList*` `resultList`,
 `tibasTupleList` `oldValueList`)

Purpose Compares values in a list of specified values with the values in the existing tuples. If value matches are found, takes the specified tuples.

Parameters	Parameter	Description
	<code>space</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>resultList</code>	A list of the values in the space.
	<code>oldValueList</code>	A list of old values. If you specify NULL in a list item, only empty tuples are taken.

Remarks Use the `tibasSpace_CompareAndTakeAll()` function to look for tuples containing specified values, and if value matches are found, take the tuple values. If there is no mismatch, the function returns the same value as specified in the list of old values.

See Also `tibasSpace_CompareAndTake()`, `tibasSpace_CompareAndTakeEx()`, `tibasSpace_CompareAndTakeAllEx()`

tibasSpace_CompareAndTakeAllEx()

Function

Declaration `tibas_status tibasSpace_CompareAndTakeAllEx(
 tibasSpace space,
 tibasSpaceResultList* resultList,
 tibasTupleList oldValueList,
 tibasTakeOptions options)`

Purpose Compares values in a list of specified values with the values in the existing tuples. If value matches are found, takes the specified tuples. Allows you to set lock and lock wait attributes for the tuples.

Parameters

Parameter	Description
space	The TIBCO ActiveSpaces entity on which the function is invoked.
resultList	A list of the values in the space.
oldValueList	A list of old values. If you specify NULL in a list item, only empty tuples are replaced.
options	Provide a tibasTakeOptions structure that specifies the options for the take.

Remarks Use the tibasSpace_CompareAndTakeAllEx() function to look for tuples containing specified values, and if value matches are found, take the tuple values. If there is no mismatch, the function returns the same value as specified in the list of old values.

tibasSpace_CompareAndTakeEx() also allows you to specify additional options for the operation, by passing a tibasTakeOptions structure in the options parameter. The tibasTakeOptions structure is defined as follows:

```
struct _tibasTakeOptions {
    tibas_long    lockWait;
    tibas_boolean lock;
    tibas_boolean unlock;
    tibas_boolean forget;
};
```

The take options are defined as follows:

- **lockWait** Specifies a lock wait time, in milliseconds.
- **lock** Take the data and lock the tuple.

- **unlock** Unlock the tuple. Only the thread or process that locked the tuple can unlock it.

Lock and unlock are mutually exclusive. If you use one of these, code NULL for the other.

- **forget** Specify this option if you want the system to forget the return. This is useful if you do not need the tuple.

tibasSpace_Listen()

Function

Declaration `tibas_status` `tibasSpace_Listen`
 (`tibasSpace` `space`,
 `tibasListener` `listener`,
 `tibasListenerDef` `listenerDef`,
 `const char*` `filter`)

Purpose Attaches a listener to a space for receiving space events.

Parameters	Parameter	Description
	<code>space</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>listener</code>	The listener object returned by the function.
	<code>listenerDef</code>	The listener definition.
	<code>filter</code>	A filter string to apply to the listener or NULL.

Remarks Use the `tibasSpace_Listen()` function to activate a `tibasListener` object for a specified space and initialize a listener for the space.

Before you can activate the listener, you must be connected to the space on which you want to listen for events; and you must get the parameters for the call to `tibasSpaceListen()` from calls to several additional ActiveSpaces functions:

- **space** The Space object that is returned by calling `tibasMetaspace_GetSpace()` or `tibasMetaspace_GetSpaceEx()`.
- **listener** A `tibasListener` object that is returned by calling the `tibasListener_Create()` function.
- **listenerDef** Specifies a `listenerDef` object that is returned by calling `tibasListenerDef_Create()` or `tibasListenerDef_CreateEx()`.

The optional last parameter, the `filter` parameter, can be used to provide a filter string that limits the events processed by the listener.

See Also `tibasMetaspace_Listen()`,`tibasListenerDef_Create()`, `tibasListenerDef_CreateEx()`

tibasSpace_Put()

Function

Declaration `tibas_status` `tibasSpace_Put`(
 `tibasSpace` `space`,
 `tibasTuple*` `oldValue`,
 `tibasTuple` `newValue`)

Purpose Stores a tuple into the space and returns the corresponding tuple.

Parameters	Parameter	Description
	<code>space</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>oldValue</code>	Returns the old value of the tuple, if a value exists.
	<code>newValue</code>	The new value to be assigned to the tuple.

Remarks Use the `tibasSpace_Put()` function to store a tuple into a space and return the original value of the tuple, if it has a value.

See Also [tibasSpace_PutEx\(\)](#), [tibasSpace_PutAll\(\)](#), [tibasSpace_CompareAndPutAllEx\(\)](#), [tibasSpace_CompareAndPut\(\)](#), [tibasSpace_CompareAndPutEx\(\)](#), [tibasSpace_CompareAndPutAll\(\)](#), [tibasSpace_CompareAndPutAllEx\(\)](#)

tibasSpace_PutEx()

Function

Declaration `tibas_status tibasSpace_PutEx(
 tibasSpace space,
 tibasTuple* oldValue,
 tibasTuple newValue,
 tibasPutOptions options)`

Purpose Stores a tuple into the space and returns the corresponding tuple, and allows you to specify a lock wait value, lock or unlock value, and a forget value.

Parameters	Parameter	Description
	space	The TIBCO ActiveSpaces entity on which the function is invoked.
	oldValue	A tuple containing the old value of the tuple.
	newValue	The tuple containing key fields, which specifies the new value.
	options	Provide a tibasPutOptions structure that specifies the options for the put.

Remarks Use the `tibasSpace_PutEx()` function to store a tuple into a space and return the original value of the tuple, if it has a value. The function also allows you to specify a lock wait value, lock or unlock value, and a forget value.

The options parameter specifies a `tibasPutOptions` structure that specifies options for the put. The `tibasPutOptions` structure is defined as follows:

```
struct _tibasPutOptions {  
    tibas_long      entryTTL;  
    tibas_long      lockWait;  
    tibas_boolean  lock;  
    tibas_boolean  unlock;  
    tibas_boolean  forget;  
};
```

The put options are defined as follows:

- **entryTTL** Allows you to define the entry TTL for the tuple that is being put.
- **lockWait** Specifies a lock wait time, in milliseconds.
- **lock** Put the data and lock the tuple.
- **unlock** Unlock the tuple. Only the thread or process that locked the tuple can unlock it.

Lock and unlock are mutually exclusive. If you use one of these, code NULL for the other.

- **forget** Specify this option if you want the system to forget the return. This is useful if you do not need the tuple

See Also [tibasSpace_Put\(\)](#), [tibasSpace_PutAll\(\)](#), [tibasSpace_CompareAndPutAllEx\(\)](#), [tibasSpace_CompareAndPut\(\)](#), [tibasSpace_CompareAndPutEx\(\)](#), [tibasSpace_CompareAndPutAll\(\)](#), [tibasSpace_CompareAndPutAllEx\(\)](#)

tibasSpace_PutAll()

Function

Declaration	<code>tibas_status</code> <code>tibasSpace_PutAll</code> (<code>tibasSpace</code> <code>space</code> , <code>tibasSpaceResultList*</code> <code>resultList</code> , <code>tibasTupleList</code> <code>tupleList</code>)								
Purpose	The batch version of the put operation. Stores all of the tuples in the provided list and returns a list of results.								
Parameters	<table><tr><th>Parameter</th><th>Description</th></tr><tr><td><code>space</code></td><td>The TIBCO ActiveSpaces entity on which the function is invoked.</td></tr><tr><td><code>resultList</code></td><td>A <code>resultList</code> object containing a list of results for the put operations.</td></tr><tr><td><code>tupleList</code></td><td>A list of tuples to put in the space.</td></tr></table>	Parameter	Description	<code>space</code>	The TIBCO ActiveSpaces entity on which the function is invoked.	<code>resultList</code>	A <code>resultList</code> object containing a list of results for the put operations.	<code>tupleList</code>	A list of tuples to put in the space.
Parameter	Description								
<code>space</code>	The TIBCO ActiveSpaces entity on which the function is invoked.								
<code>resultList</code>	A <code>resultList</code> object containing a list of results for the put operations.								
<code>tupleList</code>	A list of tuples to put in the space.								
Remarks	<p>Use the <code>tibasSpace_PutAll()</code> function to put the tuples in a provided list in a specified space and return a list of results.</p> <p>The <code>resultList</code> parameter returns a result lists that contains a list of the results of the put operations. You can use the <code>tibasResult</code> operations (see Chapter 20, SpaceResult and InvokeResult) and the <code>tibasResultList</code> operations (see Chapter 21, SpaceResultList and InvokeResultList) to process the <code>resultList</code> objects and the result objects.</p>								
See Also	tibasSpace_Put() , tibasSpace_PutAllEx() , tibasSpace_CompareAndPut() , tibasSpace_CompareAndPutEx() , tibasSpace_CompareAndPutAll() , tibasSpace_CompareAndPutAllEx() , tibasSpace_LoadAll()								

tibasSpace_PutAllEx()

Function

Declaration

```
tibas_status tibasSpace_PutAllEx(  
    tibasSpace space,  
    tibasSpaceResultList* resultList,  
    tibasTupleList valueList,  
    tibasPutOptions options)
```

Purpose

The enhanced batch version of the put operation. Stores all of the tuples in the provided list and returns a list of results. Allows you to specify a lock wait value, a lock or unlock value, and a forget value.

Parameters

Parameter	Description
space	The TIBCO ActiveSpaces entity on which the function is invoked.
resultList	The list of results for the operations included in the batch.
valueList	A list of tuples to put in the space.
options	Provide a tibasPutOptions structure that specifies the options for the put.

Remarks

Use the `tibasSpace_PutAllEx()` function to put the tuples in a provided list in a specified space and return a list of results. In addition, you can specify lock, wait, and forget options for the operation.

The put options are specified in a `tibasPutOptions` structure that specifies the options for the put. The `tibasPutOptions` structure is defined as follows:

```
struct _tibasPutOptions {  
    tibas_long    entryTTL;  
    tibas_long    lockWait;  
    tibas_boolean lock;  
    tibas_boolean unlock;  
    tibas_boolean forget;  
};
```

The put options are defined as follows:

- **entryTTL** Allows you to define the entry TTL for the tuple that is being put.
- **lockWait** Specifies a lock wait time, in milliseconds.
- **lock** Put the data and lock the tuple.
- **unlock** Unlock the tuple. Only the thread or process that locked the tuple can unlock it.

Lock and unlock are mutually exclusive. If you use one of these, code NULL for the other.

- **forget** Specify this option if you want the system to forget the return. This is useful if you do not need the tuple.

See Also [tibasSpace_Put\(\)](#), [tibasSpace_PutEx\(\)](#),[tibasSpace_PutAll\(\)](#),
[tibasSpace_CompareAndPut\(\)](#),
[tibasSpace_CompareAndPutEx\(\)](#)[tibasSpace_CompareAndPutAll\(\)](#),
[tibasSpace_CompareAndPutAllEx\(\)](#)

tibasSpace_Take()

Function

Declaration `tibas_status` `tibasSpace_Take`
 (`tibasSpace` `space`,
 `tibasTuple*` `value`,
 `tibasTuple` `key`)

Purpose Returns and atomically removes from the space the tuple (if one exists) whose tuple key fields match the key fields of the tuple provided.

Parameters	Parameter	Description
	<code>space</code>	The Space object from which you want to take a tuple.
	<code>value</code>	Returns the tuple whose key fields match the key fields of the key provided (if the tuple exists).
	<code>key</code>	The tuple containing the key fields to look for.

Remarks Use the `tibasSpace_Take()` function to return and atomically remove from the space the tuple (if one exists) whose tuple key fields match the key fields of the tuple provided.

See Also [tibasSpace_TakeEx\(\)](#), [tibasSpace_TakeAll\(\)](#), [tibasSpace_CompareAndTakeAllEx\(\)](#)

tibasSpace_TakeEx()

Function

Declaration

```
tibas_status tibasSpace_TakeEx(
    tibasSpace space,
    tibasTuple* value,
    tibasTuple key,
    tibasTakeOptions options)
```

Purpose Returns and atomically removes from the space the tuple (if one exists) whose tuple key fields match the key fields of the tuple provided. Allows you to specify a lock wait value, a lock and unlock value, and a forget value.

Parameters

Parameter	Description
space	The TIBCO ActiveSpaces entity on which the function is invoked.
value	Returns a tuple containing the tuple.
key	A tuple containing a key to locate the tuple.
options	Provide a tibasTakeOptions structure that specifies the options for the take.

Remarks

Use the `tibasSpace_TakeEx()` function to return and atomically remove from the space the tuple (if one exists) whose tuple key fields match the key fields of the tuple provided. Allows you to specify a lock wait value, a lock and unlock value, and a forget value.

The options parameter provides a `tibasTakeOptions` structure that specifies the options for the take. The `tibasTakeOptions` structure is defined as follows:

```
struct _tibasTakeOptions {
    tibas_long lockWait;
    tibas_boolean lock;
    tibas_boolean unlock;
    tibas_boolean forget;
};
```

The take options are defined as follows:

- **lockWait** Specifies a lock wait time, in milliseconds.
- **lock** Take the data and lock the tuple.
- **unlock** Unlock the tuple. Only the thread or process that locked the tuple can unlock it.

Lock and unlock are mutually exclusive. If you use one of these, code NULL for the other.

- **forget** Specify this option if you want the system to forget the return. This is useful if you do not need the tuple.

See Also [tibasSpace_Take\(\)](#), [tibasSpace_TakeAll\(\)](#), [tibasSpace_CompareAndTakeAllEx\(\)](#), [tibasSpace_CompareAndTake\(\)](#), [tibasSpace_CompareAndTakeEx\(\)](#), [tibasSpace_CompareAndTakeAllEx\(\)](#)

tibasSpace_TakeAll()

Function

Declaration	<code>tibas_status</code> <code>tibasSpace_TakeAll</code> (<code>tibasSpace</code> <code>space</code> , <code>tibasSpaceResultList</code> <code>*resultList</code> , <code>tibasTupleList</code> <code>tupleList</code>)								
Purpose	Batch version of the take operation,. Requires a list of key value tuples to take and returns a list of results.								
Parameters	<table><tr><th>Parameter</th><th>Description</th></tr><tr><td><code>space</code></td><td>The TIBCO ActiveSpaces entity on which the function is invoked.</td></tr><tr><td><code>resultList</code></td><td>The list of tuples after it is taken from the space and returned.</td></tr><tr><td><code>tupleList</code></td><td>The list of tuples containing keys for tuples to take.</td></tr></table>	Parameter	Description	<code>space</code>	The TIBCO ActiveSpaces entity on which the function is invoked.	<code>resultList</code>	The list of tuples after it is taken from the space and returned.	<code>tupleList</code>	The list of tuples containing keys for tuples to take.
Parameter	Description								
<code>space</code>	The TIBCO ActiveSpaces entity on which the function is invoked.								
<code>resultList</code>	The list of tuples after it is taken from the space and returned.								
<code>tupleList</code>	The list of tuples containing keys for tuples to take.								
Remarks	Use the <code>tibasSpace_TakeAll()</code> function to take values from a specified list of tuples and return a list of tuples taken from the data store.								
See Also	<code>tibasSpace_Take()</code> , <code>tibasSpace_CompareAndTakeAllEx()</code> , <code>tibasSpace_CompareAndPut()</code> , <code>tibasSpace_CompareAndPutEx()</code> , <code>tibasSpace_CompareAndPutAll()</code> , <code>tibasSpace_CompareAndPutAllEx()</code>								

tibasSpace_TakeAllEx()

Function

Declaration

```
tibas_status tibasSpace_TakeAllEx(  
    tibasSpace space,  
    tibasSpaceResultList* resultList,  
    tibasTupleList keyList,  
    tibasTakeOptions options)
```

Purpose

Batch version of the take operation. Requires a list of key value tuples to take and returns a list of results. Allows you to specify a lock wait value, a lock or unlock value, and a forget value.

Parameters

Parameter	Description
space	The TIBCO ActiveSpaces entity on which the function is invoked.
resultList	The list of tuples after it is taken from the space and returned.
keyList	The list of tuples containing keys for tuples to take.
options	Provide a tibasTakeOptions structure that specifies the options for the take.

Remarks

Use the `tibasSpace_TakeAllEx()` function to take values from a specified list of tuples and return a list of tuples taken from the data store. You can also specify lock, wait, and forget options for the take operation.

The options parameter specifies a `tibasTakeOptions` structure that specifies the options for the take. The `tibasTakeOptions` structure is defined as follows:

```
struct _tibasTakeOptions {  
    tibas_long lockWait;  
    tibas_boolean lock;  
    tibas_boolean unlock;  
    tibas_boolean forget;  
};
```

The take options are defined as follows:

- **lockWait** Specifies a lock wait time, in milliseconds.
- **lock** Take the data and lock the tuple.
- **unlock** Unlock the tuple. Only the thread or process that locked the tuple can unlock it.

Lock and unlock are mutually exclusive. If you use one of these, code NULL for

the other.

- **forget** Specify this option if you want the system to forget the return. This is useful if you do not need the tuple.

See Also [tibasSpace_Take\(\)](#), [tibasSpace_TakeAll\(\)](#), [tibasSpace_CompareAndTakeAllEx\(\)](#), [tibasSpace_CompareAndTake\(\)](#), [tibasSpace_CompareAndTakeEx\(\)](#), [tibasSpace_CompareAndTakeAll\(\)](#), [tibasSpace_CompareAndTakeAllEx\(\)](#)

tibasSpace_GetSpaceDef()

Function

Declaration `tibas_status` `tibasSpace_GetSpaceDef`
 (`tibasSpace` `space`,
 `tibasSpaceDef*` `spaceDef`)

Purpose Returns the `spaceDef` object that defines a specified space.

Parameters	Parameter	Description
	<code>space</code>	Specify the Space object for which to return a <code>spaceDef</code> .
	<code>spaceDef</code>	The definition of the specified space.

Remarks Use the `tibasSpace_GetSpaceDef()` function to return the `spaceDef` object for a specified space.

See Also [tibasSpaceDef_Create\(\)](#), [tibasSpaceDef_GetName\(\)](#)

tibasSpace_GetName()

Function

Declaration `tibas_status` tibasSpace_GetName
 (`tibasSpace` space,
 char** spaceName)

Purpose Returns the space name.

Parameters

Parameter	Description
space	Specify the Space object for which to return a name.
spaceName	The name of the space. The program must not modify or free the string, as it is owned by the Space object.

Remarks Use the `tibasSpace_GetName()` function to return the name of a specified space. The `spaceName` parameter is the value that was assigned when the metaspace was joined by calling the [tibasMetaspace_GetSpace\(\)](#) function or the [tibasMetaspace_GetSpaceEx\(\)](#) function.

See Also [tibasMetaspace_GetSpace\(\)](#), [tibasMetaspace_GetSpaceEx\(\)](#)

tibasSpace_GetMetaspaceName()

Function

Declaration `tibas_status` tibasSpace_GetMetaspaceName(
 `tibasSpace` space,
 `char**` metaspaceName)

Purpose Returns the metaspace name for the metaspace to which a specified space belongs..

Parameters	Parameter	Description
	space	The space for which a metaspace name is to be returned.
	metaspaceName	Returns the metaspace name.

Remarks Use the tibasSpace_GetMetaspaceName() function to determine the metaspace name for the metaspace to which a specified belongs.

See Also [tibasSpace_GetMetaspace\(\)](#)

tibasSpace_GetMetaspace()

Function

Declaration `tibas_status` tibasSpace_GetMetaspace(
 `tibasSpace` space,
 `tibasMetaspace*` metaspace);

Purpose Returns the metaspace to which a specified space belongs.

Parameters	Parameter	Description
	space	Specify the space for which to return the metaspace.
	metaspace	Returns the metaspace to which the specified metaspace belongs.

Remarks Use the tibasSpace_GetMetaspace() fuinction to return the metaspace to which a specified space belongs.

See Also [tibasSpace_GetMetaspaceName\(\)](#)

tibasSpace_Size()

Function

Declaration `tibas_status` `tibasSpace_Size`
 (`tibasSpace` `space`,
 `tibas_long*` `size`,
 `const char*` `filter`)

Purpose Returns the number of entries in the space.

Parameters	Parameter	Description
	<code>space</code>	The Space object for which to get the size value.
	<code>size</code>	Pointer to a variable that will return the size value (such as the number of tuples matching the filter).
	<code>filter</code>	A sting specifying filter criteria. Can be NULL if no filter is desired.

Remarks Use the `tibasSpace_Size()` function to return the number of entries in a specified space.

You can specify a string value with the `filter` parameter that requests return of the number of entries in the space which match a given filter. If you do not provide a filter string, code NULL for the `filter` parameter.

tibasSpace_Lock()

Function

Declaration `tibas_status` `tibasSpace_Lock`
 (`tibasSpace` `space`,
 `tibasTuple*` `value`,
 `tibasTuple` `key`)

Purpose Locks the tuple associated with a specified key value.

Parameters	Parameter	Description
	<code>space</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>value</code>	Returns either the tuple containing the locked tuple or NULL if no matching tuple can be found in the space.
	<code>key</code>	The tuple containing key fields

Remarks Use the `tibasSpace_Lock()` function to lock a tuple that has a specified key value.

See Also [tibasSpace_LockEx\(\)](#), [tibasSpace_LockAll\(\)](#), [tibasSpace_LockAllEx\(\)](#),
 [tibasSpace_Unlock\(\)](#)

tibasSpace_LockEx()

Function

Declaration `tibas_status` `tibasSpace_LockEx`(
 `tibasSpace` `space`,
 `tibasTuple*` `value`,
 `tibasTuple` `key`,
 `tibasLockOptions` `options`)

Purpose Locks the tuple associated with a key tuple value, and allows you to specify a wait value and a forget value.

Parameters	Parameter	Description
	<code>space</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>value</code>	Returns either the tuple containing the locked tuple or NULL if no matching tuple can be found in the space.
	<code>key</code>	A tuple containing key fields.
	<code>options</code>	Provide a <code>tibasLockOptions</code> structure that specifies the options for the lock.

Remarks Use the `tibasSpace_LockEx()` function to lock the tuple associated with a key tuple value, and also to specify a wait value and a forget value.

The `options` parameter specifies a `tibasLockOptions()` structure that specifies the options for the lock. The `tibasLockOptions` structure is defined as follows:

```
struct _tibasLockOptions {  
    tibas_long lockWait;  
    tibas_boolean forget;  
}
```

The lock options are defined as follows:

- **lockWait** Specifies a lock wait time, in milliseconds.
- **forget** Specify this option if you want the system to forget the return. This is useful if you do not need the tuple.

See Also [tibasSpace_Lock\(\)](#), [tibasSpace_LockAll\(\)](#), [tibasSpace_LockAllEx\(\)](#), [tibasSpace_Unlock\(\)](#), [tibasSpace_UnlockAll\(\)](#)

tibasSpace_LockAll()

Function

Declaration	<code>tibas_status</code> <code>tibasSpace_LockAll</code> (<code>tibasSpace</code> <code>space</code> , <code>tibasSpaceResultList*</code> <code>resultList</code> , <code>tibasTupleList</code> <code>keyList</code>);								
Purpose	Batch version of the Lock operation. Requires a list of key value tuples to lock and returns a list of the tuples that are locked.								
Parameters	<table><tr><th>Parameter</th><th>Description</th></tr><tr><td><code>space</code></td><td>The TIBCO ActiveSpaces entity on which the function is invoked.</td></tr><tr><td><code>resultList</code></td><td>The list of tuples after they are locked.</td></tr><tr><td><code>keyList</code></td><td>The list of tuples to lock.</td></tr></table>	Parameter	Description	<code>space</code>	The TIBCO ActiveSpaces entity on which the function is invoked.	<code>resultList</code>	The list of tuples after they are locked.	<code>keyList</code>	The list of tuples to lock.
Parameter	Description								
<code>space</code>	The TIBCO ActiveSpaces entity on which the function is invoked.								
<code>resultList</code>	The list of tuples after they are locked.								
<code>keyList</code>	The list of tuples to lock.								
Remarks	Use the <code>tibasSpace_LockAll()</code> function to lock a tuples whose key values are specified in a specified <code>keyList</code> .								
See Also	tibasSpace_Lock() , tibasSpace_LockEx() , tibasSpace_LockAllEx() , tibasSpace_Unlock() , tibasSpace_UnlockAll()								

tibasSpace_LockAllEx()

Function

Declaration `tibas_status` `tibasSpace_LockAllEx`(
 `tibasSpace` `space`,
 `tibasSpaceResultList*` `resultList`,
 `tibasTupleList` `keyList`,
 `tibasLockOptions` `options`)

Purpose Batch version of the lock operation. Requires a list of key value tuples to take and returns a list of results.

Parameter	Description
<code>space</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
<code>resultList</code>	The list of tuples after it is taken from the space and returned.
<code>keyList</code>	The list of tuples to lock.
<code>options</code>	Provide a <code>tibasLockOptions</code> structure that specifies the options for the lock.

Remarks Use the `tibasSpace_LockAllEx()` function to lock a tuples whose key values are specified in a specified `keyList`. You can also specify lock wait and forget options for the lock operation.

You can use the `options` parameter to provide a `tibasLockOptions` structure that specifies the options for the lock. The `tibasLockOptions` structure is defined as follows:

```
struct _tibasLockOptions {
    tibas_long lockWait;
    tibas_boolean forget;
}
```

The lock options are defined as follows:

- **lockWait** Specifies a lock wait time, in milliseconds.
- **forget** Specify this option if you want the system to forget the return. This is useful if you do not need the tuple.

See Also [tibasSpace_Lock\(\)](#), [tibasSpace_LockEx\(\)](#), [tibasSpace_LockAllEx\(\)](#), [tibasSpace_Unlock\(\)](#), [tibasSpace_UnlockAll\(\)](#)

tibasSpace_Unlock()

Function

Declaration `tibas_status` tibasSpace_Unlock
 (`tibasSpace` space,
 `tibasTuple` key)

Purpose Unlocks a tuple in a specified space that has a specified key.

Parameters

Parameter	Description
space	The Space object for the space in which you want to unlock a tuple.
key	Specifies the key value for the tuple to unlock.

Remarks Use the `tibasSpace_Unlock()` function to unlock a tuple with a specified key in a specified space

See Also [tibasSpace_UnlockAll\(\)](#), [tibasSpace_Lock\(\)](#), [tibasSpace_LockAll\(\)](#)

tibasSpace_UnlockAll()

Function

Declaration `tibas_status` `tibasSpace_UnlockAll`(
 `tibasSpace` `space`,
 `tibasSpaceResultList*` `resultList`,
 `tibasTupleList` `keyList`)

Purpose Batch version of the unlock operation. Requires a list of tuples to unlock and returns a list of results.

Parameters	Parameter	Description
	<code>space</code>	The Space object for the space in which you want to unlock tuples.
	<code>resultList</code>	The list of tuples that are unlocked.
	<code>keyList</code>	A list of key values for tuples that are to be unlocked.

Remarks Use the `tibasSpace_UnlockAll()` function to unlock tuples in a specified space that have specified key values.

See Also [tibasSpace_Unlock\(\)](#),[tibasSpace_Lock\(\)](#), [tibasSpace_LockAll\(\)](#)

tibasSpace_SetDistributionRole()

Function

Declaration	<code>tibas_status</code> <code>tibasSpace_SetDistributionRole</code> (<code>tibasSpace</code> <code>space</code> , <code>tibas_distributionRole</code> <code>role</code>)							
Purpose	Sets the distribution role of the member for the space.							
Parameters	<table><tr><th>Parameter</th><th>Description</th></tr><tr><td><code>space</code></td><td>Specifies the Space object for which to set the distribution role.</td></tr><tr><td><code>role</code></td><td>The role of the space can be either <code>TIBAS_ROLE_LEECH</code> or <code>TIBAS_ROLE_SEEDER</code>.</td></tr></table>		Parameter	Description	<code>space</code>	Specifies the Space object for which to set the distribution role.	<code>role</code>	The role of the space can be either <code>TIBAS_ROLE_LEECH</code> or <code>TIBAS_ROLE_SEEDER</code> .
Parameter	Description							
<code>space</code>	Specifies the Space object for which to set the distribution role.							
<code>role</code>	The role of the space can be either <code>TIBAS_ROLE_LEECH</code> or <code>TIBAS_ROLE_SEEDER</code> .							
Remarks	<p>Use the <code>tibasSpace_SetDistributionRole()</code> function to change the distribution role for the space after it has been set initially.</p> <p>The distribution role is set initially when your application sets the distribution role using <code>tibasMetaspace_GetSpaceEx()</code> and then joins the space by calling the <code>tibasMetaspace_Connect()</code> function and referencing the <code>MemberDef</code> object that specifies the distribution role, or when the application joins the space by calling the <code>tibasMetaspace_GetSpace()</code> function.</p>							
See Also	<code>tibasMetaspace_Connect()</code> , <code>tibasMetaspace_GetSpace()</code> , <code>tibasMetaspace_GetSpaceEx()</code>							

tibasSpace_SetPersister()

Function

```
Declaration      tibas_status tibasSpace_SetPersister
                   (tibasSpace    space,
                   tibasPersister persister)
```

Purpose	Register the persister object being specified as a persister on the space.
----------------	--

Parameters

Parameter	Description
space	The TIBCO ActiveSpaces entity on which the function is invoked.
persister	The persister object to be registered.

Remarks If you are implementing shared all persistence, use the `tibasSpace_SetPersister()` function to register a persister for a specified space.

Before you can call `tibasSpace_SetPersister()`, you must create a persister object by calling the `tibasPersister_Create()` function.

The persister object that you register specifies functions that will be called at various points in the life cycle of the space; for example, when the space is in the READY state to read data from local storage, or when a member leaves the space, to write data to local storage.

For general information on implementing shared all persistence, see the [Persistence](#) section in the *TIBCO ActiveSpaces Developer's Guide*. For general information on implementing persistence, see the [Implementing Persistence](#) section in the *Developer's Guide*.

See Also [tibasPersister_Create\(\)](#)

tibasSpace_Invoke()

Function

Declaration `tibas_status tibasSpace_Invoke(
 tibasSpace space,
 tibasInvokeResult* invokeResult,
 tibasTuple value,
 tibasInvokeOptions options;`

Purpose Invokes an appropriate function defined in the application code of a member that seeds a specified space.

Parameters	Parameter	Description
	space	The space on which the function is to be invoked.
	invokeResult	Returns the result of the invocation.
	value	A tuple that specifies the key for the operation.
	options	Specifies a <code>tibasInvokeOptions</code> structure that specifies the values used for an invoke operation.

Remarks Use the `tibasSpaceInvoke()` function to invoke a specified function on a member that acts as a seeder or for a specified key (or for a member that would be an active seeder if there is currently no tuple stored in the space for the specified key).

See Also [tibas_SetInvocable\(\)](#), [tibas_SetMemberInvocable\(\)](#), [tibasSpace_InvokeMember\(\)](#), [tibasSpace_InvokeMembers\(\)](#), [tibasSpace_InvokeRemoteMembers\(\)](#), [tibasSpace_InvokeSeeders\(\)](#), [tibasInvokeOptions_Initialize\(\)](#)

tibasSpace_InvokeMember()

Function

Declaration `tibas_status` `tibasSpace_InvokeMember`(
 `tibasSpace` `space`,
 `tibasInvokeResult*` `invokeResult`,
 `tibasMember` `member`,
 `tibasInvokeOptions` `options`)

Purpose Invokes a specified function on a specified space member.

Parameters	Parameter	Description
	<code>space</code>	The space on which the function is to be invoked.
	<code>invokeResult</code>	Returns the result of the invocation.
	<code>member</code>	Specify the member name for the member on which the function is to be invoked.
	<code>options</code>	Specifies a <code>tibasInvokeOptions</code> structure that specifies the values used for an invoke operation.

Remarks Use the `tibasSpace_InvokeMember()` function to invoke a specified application function on a specified space member.

See Also [tibasSpace_Invoke\(\)](#), [tibasSpace_InvokeMembers\(\)](#),
[tibasSpace_InvokeRemoteMembers\(\)](#), [tibasSpace_InvokeSeeders\(\)](#),
[tibasInvokeOptions_Initialize\(\)](#)

tibasSpace_InvokeMembers()

Function

Declaration

```
tibas_status tibasSpace_InvokeMembers(  
    tibasSpace space,  
    tibasInvokeResultList* invokeResultList,  
    tibasMemberList memberList,  
    tibasInvokeOptions options)
```

Purpose

Invokes a specified application function on the members of a specified space.

Parameters

Parameter	Description
space	The space on which the function is to be invoked.
invokeResultList	Returns a list of the results of the function invocations.
memberList	Specifies a tibasMemberList structure that specifies the list of members on which to invoke the function.
options	Specifies a tibasInvokeOptions structure that specifies the values used for an invoke operation.

Remarks

Use the `tibasSpace_InvokeMembers()` function to invoke a specified application function on the members of a specified space.

See Also

[tibasSpace_Invoke\(\)](#), [tibasSpace_InvokeMember\(\)](#), [tibasSpace_InvokeRemoteMembers\(\)](#), [tibasSpace_InvokeSeeders\(\)](#), [tibasInvokeOptions_Initialize\(\)](#)

tibasSpace_InvokeRemoteMembers()

Function

Declaration `tibas_status` tibasSpace_InvokeRemoteMembers(
 tibasSpace space,
 tibasInvokeResultList* invokeResultList,
 tibasInvokeOptions options)

Purpose This is a deprecated function.
 Invokes a specified application function on the members of a remote space.

Parameters	Parameter	Description
	space	The space on which the function is to be invoked.
	invokeResultList	Returns a list of the results of the function invocations.
	options	Specifies a <code>tibasInvokeOptions</code> structure that specifies the values used for an invoke operation.

Remarks Use the `tibasSpace_InvokeRemoteMembers()` function to invoke a specified application function on the members of a specified remote space.

See Also [tibasSpace_Invoke\(\)](#), [tibasSpace_InvokeMember\(\)](#), [tibasSpace_InvokeMembers\(\)](#), [tibasSpace_InvokeSeeders\(\)](#), [tibasInvokeOptions_Initialize\(\)](#)

tibasSpace_InvokeSeeders()

Function

Declaration

```
tibas_status tibasSpace_InvokeSeeders(  
    tibasSpace space,  
    tibasInvokeResultList* invokeResultList,  
    tibasInvokeOptions options)
```

Purpose

Invokes a specified application function on all seeders that have joined a specified space.

Parameters

Parameter	Description
space	The space on which the function is to be invoked.
invokeResultList	Returns a list of the results of the function invocations.
options	Specifies a tibasInvokeOptions structure that specifies the values used for an invoke operation.

Remarks

Use the tibasSpace_InvokeSeeders() function to invoke a specified application function on all of the seeders for a specified space.

See Also

[tibasSpace_Invoke\(\)](#), [tibasSpace_InvokeMember\(\)](#), [tibasSpace_InvokeRemoteMembers\(\)](#), [tibasSpace_InvokeMember\(\)](#), [tibasInvokeOptions_Initialize\(\)](#)

tibasSpace_Load()

Function

Declaration `tibas_status` `tibasSpace_Load`(
 `tibasSpace` `space`,
 `tibasTuple*` `oldValue`,
 `tibasTuple` `newValue`)

Purpose Loads a tuple into a specified space, but does not trigger a persister's `onWrite` function. Used in the `Onload` function provided to implement shared all persistence.

Parameters	Parameter	Description
	<code>space</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>oldValue</code>	The tuple returned by the function.
	<code>newValue</code>	The tuple loaded into the space by the function.

Remarks Use the `tibasSpace_Load()` function in your application's implementation of the `OnLoad` function that is used to implement shared all persistence.

The load function is the only one that can be invoked on a space that is in the loading phase. `tibasSpace_Load()` behaves exactly like a `put`, except that it does not trigger an invocation of the persister's `onWrite` function.

See Also [tibasSpace_LoadAll\(\)](#)

tibasSpace_LoadAll()

Function

Declaration `tibas_status` tibasSpace_LoadAll(
 tibasSpace space,
 tibasSpaceResultList* resultList,
 tibasTupleList valueList)

Purpose Batch version of the `tibasSpace_Load()` function. Used if shared all persistence is implemented.

Parameters	Parameter	Description
	space	Specifies the Space object for the TIBCO ActiveSpaces entity on which the function is invoked.
	resultList	The list of tuples returned by the function.
	valueList	The list of tuples loaded into the space by the function.

Remarks Use the `tibasSpace_LoadAll()` function to load the tuples specified in a list of tuples into a specified space. Typically this function is used in an applications `onLoad` function that is used to implement shared all persistence.

`tibasSpace_LoadAll()` is similar to [tibasSpace_PutAll\(\)](#); however, it does not trigger an invocation of the persister's `onWrite` function

See Also [tibasSpace_Load\(\)](#), [tibasSpace_PutAll\(\)](#)

tibasSpace_Free()

Function

```
Declaration      tibas_status tibasSpace_Free
                   (tibasSpace* space)
```

Purpose	Leaves the space. Destroys allocated resources and frees memory.
----------------	--

Parameters

Parameter	Description
space	The TIBCO ActiveSpaces entity on which the function is invoked.

Remarks Use the `tibasSpace_Free()` function to leave a specified space and free the memory and resources allocated to the space.

See Also [tibasSpace_Get\(\)](#)

tibasRecoveryOptions_Initialize()

Function

Declaration void tibasRecoveryOptions_Initialize(
 tibasRecoveryOptions* options)

Purpose Sets the values in a structure used to call the
 tibasMetaspace_RecoverSpaceEx() function.

Parameters	Parameter	Description
	options	Specifies a tibasRecoveryOptions structure that specifies the options for a tibasMetaspace_RecoverSpaceEx operation.

Remarks Use the tibasRecoveryOptions_Initialize() function to set the values used for recovering a space.

 The options parameter specifies the recovery options to be initialized. For more information, see [tibasMetaspace_RecoverSpaceEx\(\)](#), page 28.

See Also [tibasMetaspace_RecoverSpaceEx\(\)](#)

tibasPutOptions_Initialize()

Function

Declaration	void tibasPutOptions_Initialize(tibasPutOptions* options)	
Purpose	Sets the values in a structure used to call the tibasSpace_PutAllEx() function or the tibasSpace_CompareAndPutAllEx() function.	
Parameters	Parameter	Description
	Options	Specifies a tibasPutOptions structure that specifies the values used for an external put or an external compare and put.
Remarks	Use the tibasPutOptions_Initialize() function to initialize the values used to perform an external put operation or an external compare and put operation. For more information, see tibasSpace_PutAllEx() , page 90.	
See Also	tibasSpace_CompareAndPutEx() , tibasSpace_CompareAndPutEx()	

tibasTakeOptions_Initialize()

Function

Declaration	void tibasTakeOptions_Initialize(tibasTakeOptions* options);	
Purpose	Sets the values in a structure used to call the tibasSpace_TakeAllEx() function or the tibasSpace_CompareAndTakeAllEx() function.	
Parameters	Parameter	Description
	options	Specifies a tibasTakeOptions structure that specifies the values used for an external take or an external compare and take.
Remarks	Use the tibasTakeOptions_Initialize() function to initialize the values used to perform an external take operation or an external compare and take operation. For more information, see tibasSpace_TakeAllEx() , page 96 .	
See Also	tibasSpace_CompareAndTakeEx()	

tibasLockOptions_Initialize()

Function

Declaration	void tibasLockOptions_Initialize(tibasLockOptions* options)	
Purpose	Sets the values in a structure used to call the tibasSpace_LockEx() function.	
Parameters	Parameter	Description
	options	Specifies a tibasLockOptions structure that specifies the values used for an external lock operation.
Remarks	Use the tibasLockOptions_Initialize() function to initialize the values used to perform an external lock operation. For more information, see tibasSpace_LockEx() , page 104.	
See Also	tibasSpace_LockEx()	

tibasUnlockOptions_Initialize()

Function

Declaration `void tibasUnlockOptions_Initialize(
 tibasUnlockOptions* options)`

Purpose Sets the values in a `tibasUnlockOptions` structure.

Parameters	Parameter	Description
	<code>options</code>	Specifies a <code>tibasLockOptions</code> structure that sets the values used for an unlock operation.

Remarks Use the `tibasUnlockOptions_Initialize()` function to set the values in a `tibasUnlockOptions` structure.

The `tibasUnlockOptions` structure is defined as follows:

```
struct _tibasUnlockOptions {  
    tibas_long lockWait;  
    void* resultHandler;  
    void* closure;  
};
```

You can use the elements in the `tibasUnlockOptions` struct to specify a `ResultHandler` object or closure object when waiting asynchronously for a lock on a tuple, or collection of tuples, in a space. The `lockWait` element is used to specify the amount of time to wait for a lock to clear.

See Also [tibasLockOptions_Initialize\(\)](#)

tibasInvokeOptions_Initialize()

Function

Declaration	void tibasInvokeOptions_Initialize(tibasInvokeOptions* options)	
Purpose	Sets the values in a structure used to call the tibasSpace_Invoke() function or the tibasSpace_InvokeMember() function.	
Parameters	Parameter	Description
	options	Specifies a tibasInvokeOptions structure that specifies the values used for an invoke operation.
Remarks	Use the tibasInvokeOptions_Initialize() function to initialize the values used to perform a remote function invocation. For more information, see tibasSpace_Invoke() , page 111.	
See Also	tibasSpace_Invoke() , tibasSpace_InvokeMember()	

Chapter 4 **Admin**

This chapter explains the administrative tasks for TIBCO ActiveSpaces, which are performed using the TIBCO ActiveSpaces administration language commands.

Topics

- [Admin Operations](#)

Admin Operations

The following table lists the Admin operations:

Table 9 Admin

Function or Type	Description	Page
tibasAdmin_Create()	Returns a new admin manager which allows execution of the TIBCO ActiveSpaces administrative commands.	127
tibasAdmin_Execute()	Executes an TIBCO ActiveSpaces administration language command.	129
tibasAdmin_Connect()	Connects to the a specified admin manager and executes a specified admin command.	128
tibasAdmin_Free()	Destroys allocated resources and frees memory.	131

tibasAdmin_Create()

Function

```
Declaration      tibas_status  tibasAdmin_Create
                   (tibasAdmin*   admin)
```

Purpose	Returns a new admin manager which allows execution of the TIBCO ActiveSpaces administrative commands.
----------------	---

Parameters	Parameter	Description
	admin	The Admin manager returned by the function.

Remarks	Use the <code>tibasAdmin_Create()</code> function to create a new Admin manager that allows execution of TIBCO ActiveSpaces Admin commands.
----------------	---

See Also [tibasAdmin_Connect\(\)](#), [tibasAdmin_Execute\(\)](#)

tibasAdmin_Connect()

Function

Declaration

```
tibas_status tibasAdmin_Connect()  
    tibasAdmin      admin,  
    tibasMetaspace* metaspace,  
    const char*     cmd)
```

Purpose

Connects to the a specified admin manager and executes a specified admin command.

Parameters

Parameter	Description
admin	Specifies the admin manager to connect to.
metaspace	Specifies the metaspace to administer.
cmd	Specifies the admin command to execute.

Remarks

Use the `tibasAdmin_Connect()` command to connect to a specified Admin manager for a specified metaspace and execute a specified command.

The command that you execute can be any command that is supported by the TIBCO ActiveSpaces Admin utility.

For a complete list of the Admin commands, see [Administering ActiveSpaces with the Admin CLI](#) in the *TIBCO ActiveSpaces Administration* document.

See Also

[tibasAdmin_Execute\(\)](#)

tibasAdmin_Execute()

Function

Declaration `tibas_status` tibasAdmin_Execute
 (`tibasAdmin` `admin`,
 `char**` `result`,
 `tibasMetaspace` `metaspace`,
 `const char*` `cmd`)

Purpose Executes an TIBCO ActiveSpaces administration language command.

For commands such as `define space <space-name>`, the execute command may return a pointer to a string depending on the command being executed. The Metaspace on which the command should be executed must be passed in.

Parameters

Parameter	Description
<code>admin</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
<code>result</code>	A pointer to a NULL-terminated character buffer than contains the result of executing the command.
<code>metaspace</code>	The metaspace on which the command is executed.
<code>cmd</code>	The execute command.

Remarks

Use the `tibasAdmin_Execute()` function to execute an Admin command on a specified metaspace, using a specified Admin manager.

The result parameter contains a pointer to a NULL-terminated character buffer that contains the results of the command.

Before processing the result buffer:

1. Call `strlen()` to determine the size of the buffer. Based on the value returned by `strlen()`, you can allocate memory to store the result for processing.
2. Define a NULL pointer to point to the character buffer that will hold the result.

If you use a non-NULL pointer to point to the result parameter, ActiveSpaces returns an error. After you are done processing the result, free the memory used by the result by calling the `tibas_FreeData()` function.

The value for the `admin` parameter is returned when you call `tibasAdmin_Create()` to create an Admin manager.

Example The following example illustrates the recommended way to use `tibasAdminExecute()`:

```
char adminCommand[1024];
char* userMessagePtr = NULL;

sprintf(adminCommand, "help");
if (tibasAdmin_Execute(admin, &userMessagePtr, *metaspace,
adminCommand) != TIBAS_OK)
{ do some error handling... }
/* when finished with the contents of userMessagePtr, free it
*/
tibas_FreeData(&userMessagePtr);
/* if you want to re-use userMessagePtr for another admin
command, set it to NULL again */
userMessagePtr = NULL;
```

See Also [tibasAdmin_Create\(\)](#), [tibasAdmin_Connect\(\)](#), [tibas_FreeData\(\)](#)

Chapter 5 **SpaceDef**

A space definition is composed of two parts:

- A set of space attributes and policies that define the space's behavior and mode of deployment.
- A set of field definitions that describe the format of the data that will be stored in the space.

Topics

- [SpaceDef Operations](#)

SpaceDef Operations

The following table lists the SpaceDef operations:

Table 10 *SpaceDef*

Function or Type	Description	Page
tibasSpaceDef_Create()	Returns a new SpaceDef instance.	138
tibasSpaceDef_GetName()	Returns the space name for a specified Space object.	140
tibasSpaceDef_SetName()	Sets the space name for a specified spaceDef.	141
tibasSpaceDef_GetReplicationCount()	Returns the replication count.	142
tibasSpaceDef_SetReplicationCount()	Sets the replication count.	143
tibasSpaceDef_GetCapacity()	Gets the capacity (in number of tuples per seeder) for a specified space.	144
tibasSpaceDef_SetCapacity()	Sets the capacity (in number of tuples per seeder) of the SpaceDef.	145
tibasSpaceDef_SetEvictionPolicy()	Specifies the eviction policy for a specified space.	146
tibasSpaceDef_GetEvictionPolicy()	Returns the eviction policy set for a specified space.	147
tibasSpaceDef_GetLockScope()	Returns the lock scope value that is set for a specified space.	148
tibasSpaceDef_SetLockScope()	Specifies the lock scope value that is set for a specified space.	149
tibasSpaceDef_GetNumFields()	Returns the number of fields associated with the space definition.	150
tibasSpaceDef_GetIndex()	Retrieves the index fields defined for a specified index.	151
tibasSpaceDef_RemoveIndexDef()	Removes a specified index for a specified spaceDef.	152

Table 10 *SpaceDef*

Function or Type	Description	Page
<code>tibasSpaceDef_GetMinSeederCount()</code>	Gets from the SpaceDef the minimum number of seeders that has been specified for the space to be ready.	153
<code>tibasSpaceDef_SetMinSeederCount()</code>	Sets the minimum number of seeders required for the space to be ready.	154
<code>tibasSpaceDef_GetUpdateTransport()</code>	Returns the transport setting used for transmitting updates to the data for a specified space.	155
<code>tibasSpaceDef_SetUpdateTransport()</code>	Sets the type of protocol to be used for transmitting updates to the data for a specified space.	156
<code>tibasSpaceDef_SetKey()</code>	Sets as primary key for the space the set of fields named in the comma-separated string passed as the argument.	157
<code>tibasSpaceDef_SetDistributionFields()</code>	Sets specified key fields in a space definition as distribution fields.	158
<code>tibasSpaceDef_GetFieldDef()</code>	Returns the field definition for a specified field name.	159
<code>tibasSpaceDef_GetDistributionFields()</code>	Returns a list of the distribution fields that have been defined for a specified space.	160
<code>tibasSpaceDef_PutFieldDef()</code>	Associates a field definition to a specified space definition.	161
<code>tibasSpaceDef_GetFieldDefs()</code>	Returns the field definitions associated with the space definition.	162
<code>tibasSpaceDef_TakeFieldDef()</code>	Removes a field definition from a space definition.	163
<code>tibasSpaceDef_SetKeyDef()</code>	Sets the a primary key definition for a specified space.	164
<code>tibasSpaceDef_GetKeyDef()</code>	Gets the primary key definition for a specified space, including the field names and all associated attributes.	165

Table 10 *SpaceDef*

Function or Type	Description	Page
<code>tibasSpaceDef_AddIndexDef()</code>	Specifies the attributes for a secondary index to be added to the index definitions for the space.	166
<code>tibasSpaceDef_GetIndexDef()</code>	Returns the attributes of a specified secondary index that is defined for the specified space.	167
<code>tibasSpaceDef_GetIndexDefs()</code>	Returns a list of the secondary indexes defined for a specified space.	168
<code>tibasSpaceDef_IsSyncReplicated()</code>	Returns a value indicating whether a space is synchronously replicated.	169
<code>tibasSpaceDef_SetSyncReplicated()</code>	Sets replication to be performed in a synchronous manner.	170
<code>tibasSpaceDef_GetSpaceWait()</code>	Gets the space wait setting currently in effect for a specified spaceDef.	173
<code>tibasSpaceDef_SetSpaceWait()</code>	Sets the space wait setting for a specified spaceDef.	174
<code>tibasSpaceDef_GetWriteTimeout()</code>	Returns the write timeout value that is set for a specified space.	175
<code>tibasSpaceDef_SetWriteTimeout()</code>	Specifies the write timeout value for a specified space.	176
<code>tibasSpaceDef_GetReadTimeout()</code>	Returns the read timeout value for a specified space.	177
<code>tibasSpaceDef_SetReadTimeout()</code>	Specifies the read timeout value for a specified space.	178
<code>tibasSpaceDef_GetTTL()</code>	Returns the TTL (time-to-live) value for the tuples in the space.	179
<code>tibasSpaceDef_SetTTL()</code>	Sets the TTL (time-to-live) value for the tuples in the space in milliseconds.	180
<code>tibasSpaceDef_GetLockTTL()</code>	Returns the TTL (time-to-live) value of the locks on tuples in the space.	181

Table 10 *SpaceDef*

Function or Type	Description	Page
tibasSpaceDef_SetLockTTL()	Sets the number of milliseconds until locks on tuples in the space are automatically cleared.	182
tibasSpaceDef_GetLockWait()	Returns the amount of time a put, update, take, lock operations will wait for the lock to be cleared if the tuple is currently locked.	183
tibasSpaceDef_SetLockWait()	Sets the amount of time a put, update, take, or lock operations will wait for the lock to be cleared if the tuple is currently locked.	184
tibasSpaceDef_SetPersisted()	Specifies whether a specified space is persistent and uses shard all persistence.	185
tibasSpaceDef_IsPersisted()	Indicates whether a specified space is configured for persistence.	186
tibasSpaceDef_SetPersistenceType()	Sets the persistence type used for a specified space.	187
tibasSpaceDef_GetPersistenceType()	Returns the type of persistence currently set for a specified space.	188

tibasSpaceDef_Create()

Function

Declaration `tibas_status tibasSpaceDef_Create`
 (`tibasSpaceDef*` `spaceDef`)

Purpose Returns a new SpaceDef instance.

Parameters	Parameter	Description
	spaceDef	The new instance of SpaceDef returned by the function.

Remarks Use the `tibasSpaceDef_Create()` function to create a new SpaceDef instance. Once you have created the SpaceDef, you can assign the attributes of the space definition by calling additional `tibasSpaceDef` functions. For example, you can assign:

- A space name, by calling the `tibasSpaceDef_SetName()` function.
- The replication count for the space, by calling the `tibasSpaceDef_SetReplicationCount()` function.
- A distribution role for the space, by calling `tibasSpaceDef_SetDistributionPolicy()`.

After you have set up the attributes of the space, you pass the SpaceDef object to the `tibasMetaspace_DefineSpace()` function to define the space and then join the space by calling the `tibasMetaspace_GetSpace()` function or the `tibasMetaspace_GetSpaceEx()` function.

See Also See the following function reference topics for more information about defining the space and joining the space.

tibasSpaceDef Functions

`tibasSpaceDef_SetCapacity()`, `tibasSpaceDef_SetEvictionPolicy()`,
`tibasSpaceDef_SetReplicationCount()`, `tibasSpaceDef_SetKeyDef()`,
`tibasSpaceDef_SetMinSeederCount()`, `tibasSpaceDef_SetUpdateTransport()`,
`tibasSpaceDef_PutFieldDef()`, `tibasSpaceDef_AddIndexDef()`,
`tibasSpaceDef_SetSyncReplicated()`, `tibasSpaceDef_SetDistributionPolicy()`,
`tibasSpaceDef_SetTTL()`, `tibasSpaceDef_GetLockWait()`,
`tibasSpaceDef_SetPersisted()`, `tibasSpaceDef_SetPersistenceType()`,
`tibasSpaceDef_SetName()`

tibasMetaspace Functions

[tibasMetaspace_GetSpace\(\)](#), [tibasMetaspace_GetSpaceEx\(\)](#)

tibasSpaceDef_GetName()

Function

Declaration `tibas_status` `tibasSpaceDef_GetName`
 (`tibasSpaceDef` `spaceDef`,
 `char**` `spaceName`)

Purpose Returns the space name for a specified Space object.

Parameters	Parameter	Description
	<code>spaceDef</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>spaceName</code>	The name of the space returned by the function.

Remarks Use the `tibasSpaceDef_GetName()` function to return the space name defined in a specified SpaceDef object.

 Your application must not modify or free the `spaceName` string, as it is owned by the SpaceDef object.

See Also [tibasSpaceDef_SetName\(\)](#)

tibasSpaceDef_SetName()

Function

Declaration `tibas_status` tibasSpaceDef_SetName
 (`tibasSpaceDef` spaceDef,
 const char* spaceName)

Purpose Sets the space name for a specified spaceDef.

Parameters	Parameter	Description
	spaceDef	The TIBCO ActiveSpaces entity on which the function is invoked.
	spaceName	The space name.

Remarks Use the `tibasSpaceDef_SetName()` function to assign a name to the space represented by a specified SpaceDef object.

See Also [tibasSpaceDef_SetName\(\)](#)

tibasSpaceDef_GetReplicationCount()

Function

Declaration `tibas_status` `tibasSpaceDef_GetReplicationCount`
 (`tibasSpaceDef` `spaceDef`,
 `tibas_int*` `replicationCount`)

Purpose Returns the replication count.

Parameters	Parameter	Description
	<code>spaceDef</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>replicationCount</code>	The degree of replication set for the space.

Remarks Use the `tibasSpaceDef_GetReplicationCount()` function to query the degree of replication that is set for a specified space. To identify the space, provide the `SpaceDef` object that identifies the space.

 The replication count specifies how many seeders will be used to replicate data. For example, if the replication count is 1, each tuple seeded by one member of the space will be replicated by on other seeder of the space.

 The default value is 0 (no replication).

See Also [tibasSpaceDef_SetReplicationCount\(\)](#)

tibasSpaceDef_SetReplicationCount()

Function

Declaration `tibas_status` tibasSpaceDef_SetReplicationCount
 (`tibasSpaceDef` `spaceDef`,
 `tibas_int` `replicationCount`)

Purpose Sets the replication count.

Parameters	Parameter	Description
	<code>spaceDef</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>replicationCount</code>	The degree of replication. The default value is 0 (no replication).

Remarks Use the `tibasSpaceDef_SetReplicationCount()` function to specify the degree of replication for a specified space.

The replication count can be 0 (no replication) or a positive integer.

A degree of replication of 0 (the default value) means there is no replication. Thus, if one of the seeders for the space fails suddenly, the tuples that it was seeding will disappear from the space. If, instead, the seeder leaves in an orderly manner by invoking the call to leave the space or the call to disconnect from the metaspace, there is no loss of data.

A degree of replication of 1 means that each tuple seeded by one member of the space will also be replicated by one other seeder of the space. If a seeder suddenly disappears from the metaspace, the tuples that it was seeding will automatically be seeded by the nodes that were replicating them, and no data is lost.

See Also [tibasSpaceDef_GetReplicationCount\(\)](#)

tibasSpaceDef_GetCapacity()

Function

Declaration `tibas_status tibasSpaceDef_GetCapacity(
 tibasSpaceDef spaceDef,
 tibas_long* capacity)`

Purpose Gets the capacity (in number of tuples per seeder) for a specified space.

Parameters

Parameter	Description
spaceDef	The TIBCO ActiveSpaces entity on which the function is invoked.
capacity	The capacity (in tuples per seeder) of the specified space.

Remarks Use the `tibasSpaceDef_GetCapacity()` function to get the capacity that is set for a specified space.

The capacity attribute that is set in the `spaceDef` object used to join the space to the metaspace determines how much memory is used by the seeders that are managing the space. The default capacity setting is -1 (indicates there can be an infinite number of tuples per seeder).

For more information on how capacity works, see:

- [tibasSpaceDef_SetCapacity\(\)](#), page 145
- [Defining Capacity on page 63](#) of the TIBCO ActiveSpaces Developer’s Guide.

See Also [tibasSpaceDef_SetCapacity\(\)](#)

tibasSpaceDef_SetCapacity()

Function

Declaration `tibas_status tibasSpaceDef_SetCapacity(
 tibasSpaceDef spaceDef,
 tibas_long capacity)`

Purpose Sets the capacity (in number of tuples per seeder) of the SpaceDef.

Parameters

Parameter	Description
<code>spaceDef</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
<code>capacity</code>	The capacity (in tuples per seeder) to be set for the specified space.

Remarks Use the `tibasSpaceDef_SetCapacity()` function to set the capacity value for a specified `spaceDef`. The capacity value controls how much memory is used by seeders for storing and replicating tuples in the space. The capacity is expressed in number of tuples per seeder and defaults to `-1`, which means an infinite number of tuples per seeder.

If a capacity is specified, an eviction policy is used to indicate the outcome of an operation that would result in an additional tuple being seeded by a seeder that is already at capacity. The two choices for the eviction policy are **exception**, which means that the operation will fail with the appropriate exception being stored in the Result object, or **eviction**, which means that the seeder will evict another tuple using the eviction algorithm *Least Recently Used* (LRU), meaning least recently read or modified tuple) eviction algorithm.

See Also [tibasSpaceDef_GetCapacity\(\)](#)

tibasSpaceDef_SetEvictionPolicy()

Function

Declaration `tibas_status` `tibasSpaceDef_SetEvictionPolicy`(
 `tibasSpaceDef` `spaceDef`,
 `tibas_evictionPolicy` `evictionPolicy`)

Purpose Specifies the eviction policy for a specified space.

Parameters	Parameter	Description
	<code>spaceDef</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>evictionPolicy</code>	The eviction policy to be set for the specified space.

Remarks Use the `tibasSpaceDef_SetEvictionPolicy()` function to specify the eviction policy for a specified space.

The eviction policy can be:

- **TIBAS_EVICTION_NONE** Do not evict any tuples from the space.
- **TIBAS_EVICTION_LRU** Use the least recently used (LRU) algorithm to evict tuples from the space when the capacity of the space is reached.

If you specify `TIBAS_EVICTION_LRU`, then you must also specify a capacity value for the space by calling the `tibasSpaceDef_SetCapacity()` function.

If you specify a capacity and a LRU policy, then if the space is full (the specified capacity has been reached) and a request to insert a new tuple is made, the space will evict one of the tuples to make room for the new one.

See Also `tibasSpaceDef_GetCapacity()`, `tibasSpaceDef_SetCapacity()`,
 `tibasSpaceDef_GetEvictionPolicy()`

tibasSpaceDef_GetEvictionPolicy()

Function

Declaration `tibas_status ttibasSpaceDef_GetEvictionPolicy(
 tibasSpaceDef spaceDef,
 tibas_evictionPolicy* evictionPolicy)`

Purpose Returns the eviction policy set for a specified space.

Parameters

Parameter	Description
spaceDef	The TIBCO ActiveSpaces entity on which the function is invoked.
evictionPolicy	Returns the eviction policy to be set for the specified space.

Remarks Use the `tibasSpaceDef_GetEvictionPolicy()` function to return the eviction policy that is set for a specified space.

- **TIBAS_EVICTION_NONE** Do not evict any tuples from the space.
- **TIBAS_EVICTION_LRU** Use the least recently used (LRU) algorithm to evict tuples from the space when the capacity of the space is reached.

See Also [tibasSpaceDef_SetEvictionPolicy\(\)](#)

tibasSpaceDef_GetLockScope()

Function

Declaration `tibas_status tibasSpaceDef_GetLockScope(
 tibasSpaceDef spaceDef,
 tibas_lockScope* lockScope);`

Purpose Returns the lock scope value that is set for a specified space.

Parameters

Parameter	Description
SpaceDef	Specify the spaceDef object that identifies the space for which you want to get the lock scope value.
lockScope	Returns the lock scope. Can be one of the following: <ul style="list-style-type: none">TIBAS_LOCK_SCOPE_NONETIBAS_LOCK_SCOPE_THREADTIBAS_LOCK_SCOPE_PROCESS

Remarks Use the `tibasSpaceDef_GetLockScope()` function to return the lock scope value that is set for a specified space.

Setting a lock scope prevents either threads or processes from writing over locked data.

The lock scope value can be one of the following:

- TIBAS_LOCK_SCOPE_NONE** Specifies that locks do not have a scope.
- TIBAS_LOCK_SCOPE_THREAD** Specifies that threads cannot write over data that is locked by another thread.
- TIBAS_LOCK_SCOPE_PROCESS** Specifies that processes cannot write over data that is locked by another process; however, threads within the same process can overwrite data locked by another thread in the process.

See Also [tibasSpaceDef_SetLockScope\(\)](#)

tibasSpaceDef_SetLockScope()

Function

Declaration `tibas_status tibasSpaceDef_SetLockScope(
 tibasSpaceDef spaceDef,
 tibas_lockScope lockScope);`

Purpose Specifies the lock scope value that is set for a specified space.

Parameters	Parameter	Description
	SpaceDef	Specify the spaceDef object that identifies the space for which you want to set the lock scope value.
	lockScope	Specifies the lock scope. Can be one of the following: <ul style="list-style-type: none">• TIBAS_LOCK_SCOPE_NONE• TIBAS_LOCK_SCOPE_THREAD• TIBAS_LOCK_SCOPE_PROCESS

Remarks Use the `tibasSpaceDef_SetLockScope()` function to specify the lock scope value that is set for a specified space.

Setting a lock scope prevents either threads or processes from writing over locked data.

The lock scope value can be one of the following:

- **TIBAS_LOCK_SCOPE_NONE** Specifies that locks do not have a scope.
- **TIBAS_LOCK_SCOPE_THREAD** Specifies that threads cannot write over data that is locked by another thread.
- **TIBAS_LOCK_SCOPE_PROCESS** Specifies that processes cannot write over data that is locked by another process; however, threads within the same process can overwrite data locked by another thread in the process.

See Also [tibasSpaceDef_GetLockScope\(\)](#)

tibasSpaceDef_GetNumFields()

Function

Declaration `tibas_status` `tibasSpaceDef_GetNumFields`
 (`tibasSpaceDef` `spaceDef`,
 `tibas_int*` `numFields`)

Purpose Returns the number of fields associated with the space definition.

Parameters	Parameter	Description
	<code>spaceDef</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>numFields</code>	The number of fields associated with the space definition.

Remarks Use the `tibasSpaceDef_GetNumFields()` function to return the number of fields associated with a specified space definition.

See Also [tibasSpaceDef_PutFieldDef\(\)](#)

tibasSpaceDef_GetIndex()

Function

Declaration `tibas_status tibasSpaceDef_GetIndex(
 tibasSpaceDef spaceDef,
 const char* indexName,
 tibasStringList* fieldNames)`

Purpose Retrieves the index fields defined for a specified index.

Parameters

Parameter	Description
spaceDef	The TIBCO ActiveSpaces entity on which the function is invoked.
indexName	The name of the index.
fieldNames	Returns one or more index fields, separated by commas.

Remarks Use the `tibasSpaceDef_GetIndex()` function to return a list of the index fields defined for a specified index in a specified space definition.

The `fieldNames` parameter returns a list comma-separated field names.

See Also [tibasSpaceDef_AddIndexDef\(\)](#), [tibasSpaceDef_GetIndexDef\(\)](#),
[tibasSpaceDef_GetIndexDefs\(\)](#)

tibasSpaceDef_RemoveIndexDef()

Function

Declaration `tibas_status tibasSpaceDef_RemoveIndexDef(
 tibasSpaceDef spaceDef,
 char* indexName)`

Purpose Removes a specified index for a specified spaceDef.

Parameters	Parameter	Description
	spaceDef	Specify the spaceDef object for the space definition for which you want to remove an index.
	indexName	Specify the name for the index that you want to remove.

Remarks Use the `tibasSpaceDef_RemoveIndexDef()` function to remove a specified index in a specified space definition.

After you have removed the index definition, you can use the `tibasIndexDef` functions to add a new index or to alter the fields in the index.

See Also [tibasSpaceDef_GetIndex\(\)](#)

tibasSpaceDef_GetMinSeederCount()

Function

Declaration `tibas_status tibasSpaceDef_GetMinSeederCount(
 tibasSpaceDef spaceDef,
 tibas_int seederCount))`

Purpose Gets from the SpaceDef the minimum number of seeders that has been specified for the space to be ready.

Parameters	Parameter	Description
	spaceDef	The TIBCO ActiveSpaces entity on which the function is invoked.
	seederCount	The minimum number of seeders required for the space to be ready. The default value is 1.

Remarks Use the `tibasSpaceDef_GetMinSeederCount()` function to return the minimum seeder count value for a specified space.

If this attribute of the spaceDef is defined, it means that the space will not be usable until the required number of seeders have joined it. Since it is not possible to service any operation on a space until there is at least one seeder for it, there is always an implied default value of 1.

See Also [tibasSpaceDef_SetMinSeederCount\(\)](#)

tibasSpaceDef_SetMinSeederCount()

Function

Declaration `tibas_status tibasSpaceDef_SetMinSeederCount(
 tibasSpaceDef spaceDef,
 tibas_int seederCount))`

Purpose Sets the minimum number of seeders required for the space to be ready.

Parameters	Parameter	Description
	spaceDef	The TIBCO ActiveSpaces entity on which the function is invoked.
	seederCount	The minimum number of seeders required for the space to be ready. The default value is 1.

Remarks Use the `tibasSpaceDef_SetMinSeederCount()` function to set the minimum seeder count value for a specified space.

 If this attribute of the SpaceDef is defined, it means that the space will not be usable until the required number of seeders have joined it. Since it is not possible to service any operation on a space until there is at least one seeder for it, there is always an implied default value of 1.

See Also [tibasSpaceDef_GetMinSeederCount\(\)](#)

tibasSpaceDef_GetUpdateTransport()

Function

Declaration `tibas_status` tibasSpaceDef_GetUpdateTransport(
 tibasSpaceDef spaceDef,
 tibas_updateTransport* updateTransport)

Purpose Returns the transport setting used for transmitting updates to the data for a specified space.

Parameters	Parameter	Description
	spaceDef	The TIBCO ActiveSpaces entity on which the function is invoked.
	updateTransport	Returns the value of the update transport setting for the space. Can be one of the following: <ul style="list-style-type: none">• TIBAS_TRANSPORT_UNICAST (0) Indicates that the transport method is unicast.• TIBAS_TRANSPORT_MULTICAST (1) Indicates that the transport method is IP Multicast.

Remarks Use the `tibasSpaceDef_GetUpdateTransport()` function to return the transport setting used for transmitting updates to the data for a specified space.

See Also [tibasSpaceDef_SetUpdateTransport\(\)](#)

tibasSpaceDef_SetUpdateTransport()

Function

Declaration `tibas_status` `tibasSpaceDef_SetUpdateTransport`(
 `tibasSpaceDef` `spaceDef`,
 `tibas_updateTransport` `updateTransport`)

Purpose Sets the type of protocol to be used for transmitting updates to the data for a specified space.

Parameters	Parameter	Description
	<code>spaceDef</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>updateTransport</code>	Specifies the type of transport to be used to transmit updates to the space data.

Remarks Use the `tibasSpaceDef_SetUpdateTransport()` function to return the setting for transmitting updates to the data for a specified space.

You can specify:

- `TIBAS_TRANSPORT_UNICAST` for unicast based updates
- `TIBAS_TRANSPORT_MULTICAST` for multicast based updates

See Also [tibasSpaceDef_GetUpdateTransport\(\)](#)

tibasSpaceDef_SetKey()

Function

Declaration	<code>tibas_status</code> tibasSpaceDef_SetKey(tibasSpaceDef spaceDef, const char* fieldNames)						
Purpose	Sets as primary key for the space the set of fields named in the comma-separated string passed as the argument.						
Parameters	<table><tr><th>Parameter</th><th>Description</th></tr><tr><td>spaceDef</td><td>The TIBCO ActiveSpaces entity on which the function is invoked.</td></tr><tr><td>fieldNames</td><td>The set of fields (as a comma-separated string) to be set as the primary key for the space.</td></tr></table>	Parameter	Description	spaceDef	The TIBCO ActiveSpaces entity on which the function is invoked.	fieldNames	The set of fields (as a comma-separated string) to be set as the primary key for the space.
Parameter	Description						
spaceDef	The TIBCO ActiveSpaces entity on which the function is invoked.						
fieldNames	The set of fields (as a comma-separated string) to be set as the primary key for the space.						
Remarks	<p>Use the <code>tibasSpaceDef_SetKey()</code> function to specify a primary key for a specified space.</p> <p>Specify the primary key in the <code>fieldNames</code> parameter as a list of comma-separated field values.</p>						
See Also	<code>tibasSpaceDef_SetKeyDef()</code>						

tibasSpaceDef_SetDistributionFields()

Function

Declaration `tibas_status TIBAS_COMMON_API TIBAS_CALL_API
tibasSpaceDef_SetDistributionFields(
 tibasSpaceDef c_spaceDef,
 const char* fieldNames)`

Purpose Sets specified key fields in a space definition as distribution fields.

Parameters	Parameter	Description
	c_spaceDef	A pointer to the spaceDef object for the space for which you want to set distribution fields.
	fieldNames	A comma-separated list of field names, each enclosed in quotation marks.

Remarks Use the tibasSpaceDef_SetDistributionFields() function to set specified key fields in a spaceDef as distribution fields.

Setting a key field as a distribution field specifies that any data for the specified field or fields that is identical is stored on one seeder.

See Also [tibasSpaceDef_GetDistributionFields\(\)](#)

tibasSpaceDef_GetFieldDef()

Function

Declaration `tibas_status tibasSpaceDef_GetFieldDef
(tibasSpaceDef spaceDef,
tibasFieldDef* fieldDef,
const char* fieldName)`

Purpose Returns the field definition for a specified field name.

Parameters	Parameter	Description
	spaceDef	The spaceDef object that identifies the space for which you want to get a field definition.
	fieldDef	The field definition associated with the space definition for the named field
	fieldName	The field name.

Remarks Use the `tibasSpaceDef_GetFieldDef()` function to return the field definition for a specified field in a specified space.

See Also [tibasSpaceDef_PutFieldDef\(\)](#)

tibasSpaceDef_GetDistributionFields()

Function

Declaration `tibas_status` tibasSpaceDef_GetDistributionFields(
 tibasSpaceDef c_spaceDef,
 tibasStringList* c_fieldNames)

Purpose Returns a list of the distribution fields that have been defined for a specified space.

Parameters	Parameter	Description
	c_spaceDef	A spaceDef object identifying the space for which you want to return a list of distributions fields.
	c_fieldNames	A pointer to a list of the distribution fields that are defined for the specified space.

Remarks Use the `tibasSpaceDef_GetDistributionFields()` function to return a list of the distribution fields for a specified space.

See Also [tibasSpaceDef_SetDistributionFields\(\)](#)

tibasSpaceDef_PutFieldDef()

Function

Declaration `tibas_status` tibasSpaceDef_PutFieldDef
 (`tibasSpaceDef` spaceDef,
 `tibasFieldDef` fieldDef)

Purpose Associates a field definition to a specified space definition.

Parameters	Parameter	Description
	spaceDef	The spaceDef object that identifies the space for which you want to define a field definition.
	fieldDef	The field definition.

Remarks Use the `tibasSpaceDef_PutFieldDef()` function to associate a field definition (fieldDef) with a specified space definition (spaceDef).

Before you can call the `tibasSpaceDef_PutFieldDef()` function, you must create a space definition by calling the [tibasSpaceDef_Create\(\)](#) function and create a fieldDef by calling the [tibasFieldDef_Create\(\)](#) function.

See Also [tibasSpaceDef_Create\(\)](#), [tibasFieldDef_Create\(\)](#), [tibasSpaceDef_TakeFieldDef\(\)](#)

tibasSpaceDef_GetFieldDefs()

Function

Declaration `tibas_status tibasSpaceDef_GetFieldDefs(
 tibasSpaceDef spaceDef,
 tibasFieldDefList* fieldDefList)`

Purpose Returns the field definitions associated with the space definition.

Parameters	Parameter	Description
	spaceDef	The spaceDef object that defines the space for which you want to return a list of field definitions.
	fieldDefList	Returns a field definition list for the specified space.

Remarks Use the `tibasSpaceDef_GetFieldDefs()` function to return a list of the field definitions associated with a specified `spaceDef`.

See Also [tibasSpaceDef_GetFieldDef\(\)](#), [tibasSpaceDef_PutFieldDef\(\)](#),
 [tibasSpaceDef_TakeFieldDef\(\)](#)

tibasSpaceDef_TakeFieldDef()


Function

Declaration `tibas_status` tibasSpaceDef_TakeFieldDef
 (`tibasSpaceDef` spaceDef,
 `tibasFieldDef` fieldDef)

Purpose Removes a field definition from a space definition.

Parameters	Parameter	Description
	spaceDef	The TIBCO ActiveSpaces entity on which the function is invoked.
	fieldDef	The field definition.

Remarks Use the `tibasSpaceDef_TakeFieldDef()` function to remove a field definition from a specified `spaceDef`. This is useful if you need to modify a field definition before defining the space.



The field definition cannot be changed once the space is defined. So the field definition can be removed only before the space is created.

See Also [tibasSpaceDef_GetFieldDef\(\)](#), [tibasSpaceDef_PutFieldDef\(\)](#),
[tibasSpaceDef_GetFieldDefs\(\)](#)

tibasSpaceDef_SetKeyDef()

Function

Declaration `tibas_status` `tibasSpaceDef_SetKeyDef`(
 `tibasSpaceDef` `spaceDef`,
 `tibasKeyDef` `keyDef`)

Purpose Sets the a primary key definition for a specified space.

Parameters

Parameter	Description
<code>spaceDef</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
<code>keyDef</code>	Specifies a key definition for the SpaceDef.

Remarks Use the `tibasSpaceDef_SetKeyDef()` function to set a primary key definition for a specified space.

Using a primary key with the space can dramatically speed up searches, resulting in far fewer reads to locate a tuple.

The `keyDef` parameter is a `keyDef` object that has previously been defined by calling the `tibasKeyDef_Create()` function. The `tibasKeyDefCreate()` function creates a `keyDef` object, assigns the fields used as a primary index, and by default, assigns the index for the primary key the value `TIBAS_INDEX_HASH`. You can use the `tibasKeyDef_SetIndexType()` function to change the index to `TIBAS_INDEX_TREE` if you want to use a tree index.

See Also `tibasSpaceDef_Create()`, `tibasKeyDef_Create()`, `tibasKeyDef_GetIndexType()`, `tibasKeyDef_SetIndexType()`,

tibasSpaceDef_GetKeyDef()

Function

Declaration	<code>tibas_status tibasSpaceDef_GetKeyDef(tibasSpaceDef spaceDef, tibasKeyDef* keyDef)</code>							
Purpose	Gets the primary key definition for a specified space, including the field names and all associated attributes.							
Parameters	<table><tr><th>Parameter</th><th>Description</th></tr><tr><td>spaceDef</td><td>Specifies the SpaceDef for the space.</td></tr><tr><td>keyDef</td><td>Returns the key definition for the space.</td></tr></table>		Parameter	Description	spaceDef	Specifies the SpaceDef for the space.	keyDef	Returns the key definition for the space.
Parameter	Description							
spaceDef	Specifies the SpaceDef for the space.							
keyDef	Returns the key definition for the space.							
Remarks	Use the <code>tibasSpaceDef_GetKeyDef()</code> function to return the key definition in effect for a specified space.							
See Also	tibasSpaceDef_SetKeyDef()							

tibasSpaceDef_AddIndexDef()

Function

Declaration

`tibas_status` `tibasSpaceDef_AddIndexDef(tibasSpaceDef spaceDef, tibasIndexDef indexDef)`

Purpose

Specifies the attributes for a secondary index to be added to the index definitions for the space.

Parameters

Parameter	Description
<code>spaceDef</code>	Specifies the <code>spaceDef</code> for the space.
<code>indexDef</code>	A <code>tibasIndexDef</code> structure that specifies the attributes for the secondary index.

Remarks

Use the `tibasSpaceDef_AddIndexDef()` function to add a secondary index for a specified space.

Indexes can speed up the filtering of data when setting up queries and browser filters. When used with shared-nothing persistence, indexes are particularly useful, because they return all matching tuples, including those that have been previously evicted from the space but are stored in persistent storage.

To set up an index definition, you must:

1. Create an `IndexDef` object by calling the `tibasIndexDef_Create()` function and assign the field names used for the index.
2. Call `tibasSpaceDef_AddIndexDef()` and specify the `IndexDef` object that was created by the call to `tibasIndexDef_Create()`.

See Also

`tibasSpaceDef_GetIndexDef()`, `tibasSpaceDef_GetIndexDefs()`, `tibasIndexDef_Create()`, `tibasIndexDef_SetName()`, `tibasIndexDef_SetIndexType()`, `tibasIndexDef_SetFieldNames()`

tibasSpaceDef_GetIndexDef()

Function

Declaration

```
tibas_status tibasSpaceDef_GetIndexDef(  
    tibasSpaceDef  spaceDef,  
    tibasIndexDef* indexDef,  
    char*          indexName)
```

Purpose

Returns the attributes of a specified secondary index that is defined for the specified space.

Parameters

Parameter	Description
spaceDef	Specifies the SpaceDef for the space.
indexDef	Returns a tibasIndexDef structure that contains the attributes for the secondary index.
indexName	Specify the name of the secondary index for which you want to return an indexDef.

Remarks

Use the `tibasSpaceDef_GetIndexDef()` function to return the attributes of a specified secondary index that is defined for a specified space.

See Also

[tibasSpaceDef_AddIndexDef\(\)](#), [tibasSpaceDef_GetIndexDefs\(\)](#)

tibasSpaceDef_GetIndexDefs()

Function

Declaration `tibas_status tibasSpaceDef_GetIndexDefs(
 tibasSpaceDef spaceDef,
 tibasIndexDefList* indexDefList)`

Purpose Returns a list of the secondary indexes defined for a specified space.

Parameters	Parameter	Description
	spaceDef	Specifies the SpaceDef for the space.
	indexDefList	A pointer to the list of IndexDefs that is returned by the function.

Remarks Use the `tibasSpaceDef_GetIndexDefs()` function to return a list of the secondary indexes currently defined for a specified space.

See Also [tibasSpaceDef_AddIndexDef\(\)](#), [tibasSpaceDef_GetIndexDef\(\)](#)

tibasSpaceDef_SetSyncReplicated()

Function

Declaration `tibas_status` `tibasSpaceDef_SetSyncReplicated`
 (`tibasSpaceDef` `spaceDef`,
 `tibas_boolean` `syncReplicated`)

Purpose Sets replication to be performed in a synchronous manner.

Parameters

Parameter	Description
<code>spaceDef</code>	A <code>spaceDef</code> object that specifies the space for which you want to query the type of replication that is configured.
<code>syncReplicated</code>	The default is false. To set replication to synchronous replication, code 1 (TRUE).

Remarks Use the `tibasSpaceDef_SetSyncReplicated()` function to specify whether replication for a specified space is set to synchronous replication.

See Also [tibasSpaceDef_IsSyncReplicated\(\)](#)

tibasSpaceDef_GetDistributionPolicy()

Function

[illegible]

Purpose Returns the distribution policy that is set for the space.

Parameters

Parameter	Description
spaceDef	The TIBCO ActiveSpaces entity on which the function is invoked.
distributedPolicy	The distribution policy.

Remarks	Use the <code>tibasSpaceDef_GetDistributionPolicy()</code> function to specify the distribution policy attribute of the <code>spaceDef</code> for a space.
----------------	--

The distribution policy can be one of the following:

- **TIBAS_DISTRIBUTED** (Default setting) management of the space entries is shared among the seeders that have joined the space.
- **TIBAS_NON_DISTRIBUTED** A single seeder is responsible for all tuples in the space.



Although in a nondistributed space, one seeder is responsible for the tuples, other seeders joined to the space may also replicate the tuples if a degree of replication is specified

See Also [tibasSpaceDef_SetDistributionPolicy\(\)](#)

tibasSpaceDef_SetDistributionPolicy()

Function

[illegible]

Purpose Sets the distribution policy attribute for a specified spaceDef.

Parameters

Parameter	Description
spaceDef	The spaceDef object used to define the space.ActiveSpaces entity on which the function is invoked.
distributedPolicy	The distribution policy.

Remarks	Use the <code>tibasSpaceDef_SetDistributionPolicy()</code> function to set the distribution policy attribute of the <code>spaceDef</code> for a space.
----------------	--

The distribution policy can be one of the following:

- **TIBAS_DISTRIBUTED** (Default setting) management of the space entries is shared among the seeders that have joined the space.
- **TIBAS_NON_DISTRIBUTED** A single seeder is responsible for all tuples in the space.



Although in a nondistributed space, one seeder is responsible for the tuples, other seeders joined to the space may also replicate the tuples if a degree of replication is specified

See Also [tibasSpaceDef_GetDistributionPolicy\(\)](#)

tibasSpaceDef_GetSpaceWait()

Declaration `tibas_status` tibasSpaceDef_GetSpaceWait(
 tibasSpaceDef spaceDef,
 tibas_long* spaceWait)

Purpose Gets the space wait setting currently in effect for a specified spaceDef.

Parameters	Parameter	Description
	spaceDef	Specify the spaceDef for the space for which you want to obtain the space wait setting.
	spaceWait	Returns the space wait interval in milliseconds.

Remarks Use the `tibasSpaceDef_GetSpaceWait()` function to return the space wait that is currently set for a specified space.

The space wait value is a timeout that applies to operations that cannot be processed because the space is not in the READY state, i.e., the space in the INITIAL, LOADING, RECOVER, or SUSPEND state.

See Also [tibasSpaceDef_SetSpaceWait\(\)](#), [tibasSpaceDef_GetWriteTimeout\(\)](#), [tibasSpaceDef_SetWriteTimeout\(\)](#), [tibasSpaceDef_GetReadTimeout\(\)](#), [tibasSpaceDef_SetReadTimeout\(\)](#)

tibasSpaceDef_SetSpaceWait()

Declaration `tibas_status` `tibasSpaceDef_SetSpaceWait`(
 `tibasSpaceDef` `spaceDef`,
 `tibas_long` `spaceWait`)

Purpose Sets the space wait setting for a specified `spaceDef`.

Parameters	Parameter	Description
	<code>spaceDef</code>	Specify the <code>spaceDef</code> for the space for which you want to set the space wait setting.
	<code>spaceWait</code>	Specifies the space wait interval in milliseconds.

Remarks Use the `tibasSpaceDef_GetSpaceWait()` function to specify the space wait for a specified space.

The space wait value is a timeout that applies to operations that cannot be processed because the space is not in the READY state, i.e., the space in the INITIAL, LOADING, RECOVER, or SUSPEND state.

See Also [tibasSpaceDef_GetSpaceWait\(\)](#), [tibasSpaceDef_GetWriteTimeout\(\)](#), [tibasSpaceDef_SetWriteTimeout\(\)](#), [tibasSpaceDef_GetReadTimeout\(\)](#), [tibasSpaceDef_SetReadTimeout\(\)](#)

tibasSpaceDef_GetWriteTimeout()

Function

Declaration `tibas_status tibasSpaceDef_GetWriteTimeout(
 tibasSpaceDef spaceDef,
 tibas_long* writeTimeout);`

Purpose Returns the write timeout value that is set for a specified space.

Parameters

Parameter	Description
spaceDef	Specify the spaceDef for the space for which you want to obtain the write timeout setting.
writeTimeout	Returns the write timeout value, in milliseconds.

Remarks Use the `tibasSpaceDef_GetWriteTimeout()` function to return the write timeout value that is set for a specified `spaceDef`.

The write timeout value applies to Put, Take, Lock, and Unlock operations.

See Also [tibasSpaceDef_GetSpaceWait\(\)](#), [tibasSpaceDef_SetSpaceWait\(\)](#),
[tibasSpaceDef_SetWriteTimeout\(\)](#), [tibasSpaceDef_GetReadTimeout\(\)](#),
[tibasSpaceDef_SetReadTimeout\(\)](#)

tibasSpaceDef_SetWriteTimeout()

Function

Declaration `tibas_status tibasSpaceDef_SetWriteTimeout(
 tibasSpaceDef spaceDef,
 tibas_long writeTimeout);`

Purpose Specifies the write timeout value for a specified space.

Parameters	Parameter	Description
	spaceDef	Specify the spaceDef for the space for which you want to set a write timeout value.
	writeTimeout	Specify the write timeout value, in milliseconds.

Remarks Use the `tibasSpaceDef_GetWriteTimeout()` function to specify the write timeout value that is set for a specified spaceDef.

The write timeout value applies to Put, Take, Lock, and Unlock operations.

See Also [tibasSpaceDef_GetSpaceWait\(\)](#), [tibasSpaceDef_SetSpaceWait\(\)](#),
[tibasSpaceDef_GetWriteTimeout\(\)](#), [tibasSpaceDef_GetReadTimeout\(\)](#),
[tibasSpaceDef_SetReadTimeout\(\)](#)

tibasSpaceDef_GetReadTimeout()

Function

Declaration `tibas_status tibasSpaceDef_GetReadTimeout(
 tibasSpaceDef spaceDef,
 tibas_long* readTimeout);`

Purpose Returns the read timeout value for a specified space.

Parameters	Parameter	Description
	spaceDef	Specify the spaceDef for the space for which you want to return the read timeout value.
	readTimeout	Returns the read timeout value, in milliseconds.

Remarks Use the `tibasSpaceDef_GetReadTimeout()` function to return the read timeout value for a specified spaceDef.

The read timeout value applies to Get operations.

See Also [tibasSpaceDef_GetSpaceWait\(\)](#), [tibasSpaceDef_SetSpaceWait\(\)](#),
[tibasSpaceDef_GetWriteTimeout\(\)](#), [tibasSpaceDef_SetWriteTimeout\(\)](#),
[tibasSpaceDef_GetReadTimeout\(\)](#), [tibasSpaceDef_SetReadTimeout\(\)](#)

tibasSpaceDef_SetReadTimeout()

Function

Declaration `tibas_status tibasSpaceDef_SetReadTimeout(
 tibasSpaceDef spaceDef,
 tibas_long readTimeout);`

Purpose Specifies the read timeout value for a specified space.

Parameters	Parameter	Description
	spaceDef	Specify the spaceDef for the space for which you want to set a read timeout value.
	readTimeout	Specify the read timeout value, in milliseconds.

Remarks Use the `tibasSpaceDef_GetReadTimeout()` function to specify the read timeout value for a specified spaceDef.

The read timeout value applies to Get operations.

See Also [tibasSpaceDef_GetSpaceWait\(\)](#), [tibasSpaceDef_SetSpaceWait\(\)](#),
[tibasSpaceDef_GetWriteTimeout\(\)](#), [tibasSpaceDef_SetWriteTimeout\(\)](#),
[tibasSpaceDef_GetReadTimeout\(\)](#)

tibasSpaceDef_GetTTL()

Function

Declaration `tibas_status` tibasSpaceDef_GetTTL
 (`tibasSpaceDef` `spaceDef`,
 `tibas_long*` `ttl`)

Purpose Returns the TTL (time-to-live) value for the tuples in the space.

Parameters	Parameter	Description
	<code>spaceDef</code>	A <code>spaceDef</code> object that identifies the space for which to get the TTL.
	<code>ttl</code>	The TTL in milliseconds.

Remarks Use the `tibasSpaceDef_GetTTL()` function to return the TTL for a specified space.

See Also [tibasSpaceDef_SetTTL\(\)](#)

tibasSpaceDef_SetTTL()

Function

Declaration `tibas_status` `tibasSpaceDef_SetTTL`
 (`tibasSpaceDef` `spaceDef`,
 `tibas_long` `ttl`)

Purpose Sets the TTL (time-to-live) value for the tuples in the space in milliseconds.

Parameters	Parameter	Description
	<code>spaceDef</code>	A <code>spaceDef</code> object that identifies the space for which to set the TTL.
	<code>ttl</code>	The TTL in milliseconds. The default is <code>TIBAS_WAIT_FOREVER</code> (-1).

Remarks Use the `tibasSpaceDef_SetTTL()` function to set the TTL for a specified space.

The TTL value that you specify determines how long (in milliseconds) a tuple is retained in the in-memory data grid after it was created initially or modified before it is expired.

See Also [tibasSpaceDef_SetTTL\(\)](#)

tibasSpaceDef_GetLockTTL()

Function

Declaration `tibas_status` tibasSpaceDef_GetLockTTL
 (`tibasSpaceDef` spaceDef,
 `tibas_long*` lockTTL)

Purpose Returns the TTL (time-to-live) value of the locks on tuples in the space.

Parameters	Parameter	Description
	spaceDef	A spaceDef object that identifies the space for which to return the lock TTL.
	lockTTL	The TTL in milliseconds.

Remarks Use the `tibasSpaceDef_GetLockTTL()` function to return the lock TTL for a specified space.

See Also [tibasSpaceDef_SetLockTTL\(\)](#)

tibasSpaceDef_SetLockTTL()

Function

Declaration `tibas_status tibasSpaceDef_SetLockTTL`
 (`tibasSpaceDef spaceDef,`
 `tibas_long lockTTL`)

Purpose Sets the number of milliseconds until locks on tuples in the space are automatically cleared.

Parameters

Parameter	Description
spaceDef	A spaceDef object that identifies the space for which to set the lock TTL.
lockTTL	Number of milliseconds until the lock is automatically cleared. The default is TIBAS_WAIT_FOREVER (-1).

Remarks Use the `tibasSpaceDef_SetLockTTL()` function to specify an optional TTL value for locks that are placed on tuples in the space.

By default, unless you specify a lock TTL, tuples remain locked until the application that locked the tuple clears the lock, a lock wait time set by an application is reached, or the application disconnects from the metaspace.

To prevent possible deadlock scenarios, you can call `tibasSpaceDefSetLockTTL()` to specify a lock wait time. If an application does not clear a lock that it has placed on a tuple within the specified lock TTL period, the system automatically clears the lock.

See Also [tibasSpaceDef_GetLockTTL\(\)](#)

tibasSpaceDef_GetLockWait()

Function

Declaration	<code>tibas_status</code> <code>tibasSpaceDef_GetLockWait</code> (<code>tibasSpaceDef</code> <code>spaceDef</code> , <code>tibas_long*</code> <code>lockWait</code>)	
Purpose	Returns the amount of time a put, update, take, lock operations will wait for the lock to be cleared if the tuple is currently locked.	
Parameters	Parameter	Description
	<code>spaceDef</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>lockWait</code>	Number of milliseconds until lock is automatically cleared.
		If no lock wait is set, returns <code>TIBAS_WAIT_FOREVER</code> (-1).
Remarks	Use the <code>tibasSpaceDef_GetLockWait()</code> function to return the amount of time a put, update, take, or lock operation will wait for a lock to be cleared if the tuple is currently locked.	
See Also	tibasSpaceDef_SetLockWait()	

tibasSpaceDef_SetLockWait()

Function

Declaration `tibas_status` `tibasSpaceDef_SetLockWait`
 (`tibasSpaceDef` `spaceDef`,
 `tibas_long` `lockWait`)

Purpose Sets the amount of time a put, update, take, or lock operations will wait for the lock to be cleared if the tuple is currently locked.

Parameters	Parameter	Description
	<code>spaceDef</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>lockWait</code>	Number of milliseconds until lock is automatically cleared. The default is <code>TIBAS_WAIT_FOREVER</code> (-1).

Remarks Use the `tibasSpaceDef_SetLockWait()` function to set the amount of time a put, update, take, or lock operations will wait for the lock to be cleared if the tuple is currently locked.

See Also [tibasSpaceDef_GetLockWait\(\)](#)

tibasSpaceDef_SetPersisted()

Function

Declaration `tibas_status tibasSpaceDef_SetPersisted(
 tibasSpaceDef spaceDef,
 tibas_boolean persisted)`

Purpose Specifies whether a specified space is persistent and uses shard all persistence.

Parameters

Parameter	Description
spaceDef	The TIBCO ActiveSpaces entity on which the function is invoked.
persisted	Whether or not the space is persistent. If you set this argument to <code>TIBAS_TRUE</code> , the persistence type for the space is set to shared all persistence (<code>TIBAS_PERSISTENCE_SHARED_ALL</code>).

Remarks Use the `tibasSpaceDef_SetPersisted()` function to specify whether the specified space uses shared all persistence.

If you specify `TIBAS_TRUE` for the `persisted` parameter, the space is persistent and will need to have at least one persister registered in order to be ready. The default is `TIBAS_FALSE`.

If you set the `persisted` argument to **`TIBAS_TRUE`**, the persistence type for the space is set to shared all persistence (`TIBAS_PERSISTENCE_SHARED_ALL`).

To set up shared all persistence, you must create and register at least one persister—a space member that is responsible for saving data to local storage.

To create a persister, you call the [tibasPersister_Create\(\)](#) function. This function creates a Persister object, which specifies functions provided by your application to read or write data depending on what state the space is in. You then register the persister by calling the [tibasSpace_SetPersister\(\)](#) function.

For more information on implementing shared all persistence, see [Setting up Shared All Persistence on page 77](#) of the *TIBCO ActiveSpaces Developer's Guide*.

See Also [tibasSpace_SetPersister\(\)](#), [tibasPersister_Create\(\)](#)

tibasSpaceDef_IsPersisted()

Function

Declaration `tibas_status` tibasSpaceDef_IsPersisted(`tibasSpaceDef` spaceDef)

Purpose Indicates whether a specified space is configured for persistence.

Parameters	Parameter	Description
	spaceDef	A spaceDef object that references the space whose persistence setting you want to query.

Remarks Use the `tibasSpaceDef_IsPersisted()` function to determine if a space is configured for persistence.

 The function returns `TIBAS_TRUE` if the space is persistent, and `TIBAS_FALSE` otherwise.

See Also `tibasSpaceDef_SetPersisted()`, `tibasSpaceDef_SetPersistenceType()`,

tibasSpaceDef_SetPersistenceType()

Function

Declaration `tibas_status tibasSpaceDef_SetPersistenceType(
 tibasSpaceDef spaceDef,
 tibas_persistenceType persistenceType)`

Purpose Sets the persistence type used for a specified space.

Parameters

Parameter	Description
spaceDef	A spaceDef object that references the space for which you want to set the persistence type.
persistenceType	Specifies the persistence type for the space. Specify one of the following: <ul style="list-style-type: none">• TIBAS_PERSISTENCE_NONE• TIBAS_PERSISTENCE_SHARED_ALL• TIBAS_PERSISTENCE_SHARED_NOTHING

Remarks Use the `tibasSpaceDef_SetPersistenceType()` function to specify whether a space uses persistence, and if so, the type of persistence to use.

You can specify:

- **TIBAS_PERSISTENCE_NONE** Do not persist objects in the space.
- **TIBAS_PERSISTENCE_SHARED_ALL (1)** Use shared-all persistence. With shared-all persistence, each node that joins a space as a seeder maintains a copy of the space data on disk. Each node that joins as a seeder writes its data to disk and reads the data when needed for recovery and for cache misses
- **TIBAS_PERSISTENCE_SHARED_NOTHING (2)** Use shared-nothing persistence. With shared all persistence, all nodes share a single persister or a set of persisters.

See Also [tibasSpaceDef_SetPersisted\(\)](#), [tibasSpaceDef_IsPersisted\(\)](#),
[tibasSpaceDef_GetPersistenceType\(\)](#), [tibasSpaceDef_SetPersistenceType\(\)](#)

tibasSpaceDef_GetPersistenceType()

Function

Declaration `tibas_status` tibasSpaceDef_GetPersistenceType(
 tibasSpaceDef spaceDef,
 tibas_persistenceType* persistenceType)

Purpose Returns the type of persistence currently set for a specified space.

Parameters	Parameter	Description
	spaceDef	The TIBCO ActiveSpaces entity on which the function is invoked.
	persistenceType	Returns the persistence type that is currently set for the space.

Remarks Use the `tibasSpaceDef_GetPersistenceType()` function to return the type of persistence currently set for a specified space.

The persistence type can be one of the following:

- **TIBAS_PERSISTENCE_NONE** Do not persist objects in the space.
- **TIBAS_PERSISTENCE_SHARED_ALL (1)** Use shared-all persistence. With shared-all persistence, each node that joins a space as a seeder maintains a copy of the space data on disk. Each node that joins as a seeder writes its data to disk and reads the data when needed for recovery and for cache misses
- **TIBAS_PERSISTENCE_SHARED_NOTHING (2)** Use shared-nothing persistence. With shared all persistence, all nodes share a single persister or a set of persisters.

See Also [tibasSpaceDef_SetPersisted\(\)](#), [tibasSpaceDef_IsPersisted\(\)](#),

tibasSpaceDef_Free()

Function

Declaration	<code>tibas_status</code> <code>tibasSpaceDef_Free</code> (<code>tibasSpaceDef*</code> <code>spaceDef</code>)	
Purpose	Destroys allocated resources and frees memory used by a specified space.	
Parameters	Parameter	Description
	<code>spaceDef</code>	A <code>spaceDef</code> object that references the space definition that you want to free.
Remarks	Use the <code>tibasSpaceDef_Free()</code> function to free the resources and memory used by a specified space.	
See Also	tibasSpaceDef_Create()	

Chapter 6 **MemberDef**

This chapter explains the `MemberDef` and `SpaceMemberDef` operations.

The `MemberDef` functions create and set the attributes of a `MemberDef` object that controls how a space member connects to a metaspace and communicates with other space members.

The `SpaceMemberDef` functions specify additional attributes of space membership, such as whether a member joins a space as a seeder or a leech, and, if shared-nothing persistence is implemented, the directory path in which to store persisted data.

Topics

- [MemberDef Operations, page 192](#)

MemberDef Operations

The following table lists the MemberDef operations:

Table 11 MemberDef

Function or Type	Description	Page
tibasMemberDef_Create()	Creates a connection definition to be used to connect to a metaspace.	194
tibasMemberDef_Free()	Frees a specified connection definition (memberDef).	195
tibasMemberDef_SetMemberName()	Sets the name for a specified member.	196
tibasMemberDef_SetDataStore()	Specifies the directory path for the data store used when you implement shared-nothing persistence.	197
tibasMemberDef_SetWorkerThreadCount()	Specifies the number of worker threads for a specified space number.	198
tibasMemberDef_GetWorkerThreadCount()	Returns the number of worker threads for a specified space number.	199
tibasMemberDef_GetDataStore()	Returns the directory path for data store files, if shared-nothing persistence is implemented.	201
tibasMemberDef_GetMemberName()	Returns the name for a specified member.	200
tibasMemberDef_SetRemoteDiscovery()	Sets the discovery URL for a remote member.	202
tibasMemberDef_GetRemoteDiscovery()	Gets the discovery URL for a specified remote member.	203
tibasMemberDef_SetRemoteListen()	Specifies a listen URL for a specified remote member.	205
tibasMemberDef_SetListen()	Sets the listen attribute for a specified memberDef.	206
tibasMemberDef_SetDiscovery()	Sets the discovery attribute of the specified MemberDef object.	208

Table 11 *MemberDef*

Function or Type	Description	Page
tibasMemberDef_GetListen()	Gets the listen attribute of the connection definition.	208
tibasMemberDef_GetDiscovery()	Returns the discovery attribute for a specified memberDef.	209
tibasMemberDef_SetContext()	Specifies a tuple that contains context information for a specified memberDef.	210
tibasMemberDef_GetSecurityPolicyFile()	Returns the location and filename for the policy file that is in effect for a specified member.	211
tibasMemberDef_GetSecurityTokenFile()	Returns the directory path and filename for the security token file that is in effect for a specified member.	212
tibasMemberDef_GetAuthenticationCallback()	Returns the authentication callback setting for a specified memberDef, if the callback function has been specified.	213
tibasMemberDef_SetSecurityPolicyFile()	Sets the directory path and filename for the security policy file for a specified member.	214
tibasMemberDef_SetAuthenticationCallback()	Specifies a callback routine that allows you to customize how user authentication information is retrieved for users.	216
tibasMemberDef_SetTimeout()	Specifies a timeout interval for a specified memberDef.	219

tibasMemberDef_Create()

Function

```
Declaration      tibas_status tibasMemberDef_Create
                   (tibasMemberDef* memberDef)
```

Purpose Creates a connection definition to be used to connect to a metaspace.

Parameters

Parameter	Description
memberDef	The new instance of a MemberDef object returned by the function.

Remarks	Use the <code>tibasMemberDef_Create()</code> function to create a <code>MemberDef</code> object. You can then call additional <code>tibasMemberDef</code> functions to specify the discovery URL, listen URL, and remote discovery URL for a space member.
----------------	--

See Also [tibasMemberDef_SetMemberName\(\)](#), [tibasMemberDef_GetMemberName\(\)](#), [tibasMemberDef_SetRemoteDiscovery\(\)](#), [tibasMemberDef_GetRemoteDiscovery\(\)](#), [tibasMemberDef_SetListen\(\)](#), [tibasMemberDef_SetDiscovery\(\)](#), [tibasMemberDef_GetListen\(\)](#), [tibasMemberDef_GetDiscovery\(\)](#)

tibasMemberDef_Free()

Function

```
Declaration      tibas_status tibasMemberDef_Free
                   (tibasMemberDef* memberDef)
```

Purpose	Frees a specified connection definition (memberDef).
----------------	--

Parameters

Parameter	Description
memberDef	The TIBCO ActiveSpaces entity on which the function is invoked.

Remarks	Use the <code>tibasMemberDef_Free()</code> function to free the resources used by a specified connection definition.
----------------	--

See Also [tibasMemberDef_Create\(\)](#)

tibasMemberDef_SetMemberName()

Function

Declaration `tibas_status` tibasMemberDef_SetMemberName(
 tibasMemberDef memberDef,
 const char** name)

Purpose Sets the name for a specified member.

Parameters

Parameter	Description
memberDef	The memberDef object for the member for which you want to set a name.
name	The name to set for the member.

Remarks Use the `tibasMemberDef_SetMemberName()` function to specify a name for a specified memberDef.

See Also [tibasMemberDef_Create\(\)](#), [tibasMemberDef_GetMemberName\(\)](#),
[tibasMemberDef_SetRemoteDiscovery\(\)](#), [tibasMemberDef_SetListen\(\)](#),
[tibasMemberDef_SetDiscovery\(\)](#)

tibasMemberDef_SetDataStore()

Function

Declaration `tibas_status` tibasMemberDef_SetDataStore(
 tibasMemberDef memberDef,
 const char* dataStore)

Purpose Specifies the directory path for the data store used when you implement shared-nothing persistence.

Parameters	Parameter	Description
	memberDef	Specify the memberDef object that defines the member for which you are implementing shared-nothing persistence.
	dataStore	Specifies a string that sets the directory path to the directory that holds the data store for shared-nothing persistence.

Remarks If you are implementing shared-nothing persistence, use the `tibasMemberDef_SetDataStore()` function to specify the directory where the data store for persisted data is located.

See Also [tibasMemberDef_GetDataStore\(\)](#)

tibasMemberDef_SetWorkerThreadCount()

Function

Declaration `tibas_status` tibasMemberDef_SetWorkerThreadCount(
 `tibasMemberDef` memberDef,
 `tibas_int` count);

Purpose Specifies the number of worker threads for a specified space number.

Parameters	Parameter	Description
	memberDef	Specifies the member definition for the space for which you want to set the number of worker threads.
	count	Specifies the number of worker threads.

Remarks Use the `tibasMemberDef_SetWorkerThreadCount()` function to specify the number of worker threads used with a specified space member.

See Also [tibasMemberDef_GetWorkerThreadCount\(\)](#)

tibasMemberDef_GetWorkerThreadCount()

Function

Declaration `tibas_status` tibasMemberDef_GetWorkerThreadCount(
 tibasMemberDef memberDef,
 tibas_int *count);

Purpose Returns the number of worker threads for a specified space number.

Parameters	Parameter	Description
	memberDef	Specifies the member definition for the space for which you want to return the number of worker threads.
	count	Returns the number of worker threads set for the member.

Remarks Use the `tibasMemberDef_GetWorkerThreadCount()` function to return the number of worker threads used with a specified space member.

See Also [tibasMemberDef_SetWorkerThreadCount\(\)](#)

tibasMemberDef_GetMemberName()

Function

Declaration `tibas_status tibasMemberDef_GetMemberName(
 tibasMemberDef memberDef,
 const char** name)`

Purpose Returns the name for a specified member.

Parameters	Parameter	Description
	memberDef	The memberDef object for the member for which you want to return the name.
	name	The name of the member that is returned.

Remarks Use the `tibasMemberDef_GetMemberName()` function to return the name of a specified space member.

See Also [tibasMemberDef_Create\(\)](#), [tibasMemberDef_SetMemberName\(\)](#),
[tibasMemberDef_SetRemoteDiscovery\(\)](#), [tibasMemberDef_SetListen\(\)](#),
[tibasMemberDef_SetDiscovery\(\)](#)

tibasMemberDef_GetDataStore()

Function

Declaration `tibas_status` tibasMemberDef_GetDataStore(
 tibasMemberDef memberDef,
 const char** dataStore)

Purpose Returns the directory path for data store files, if shared-nothing persistence is implemented.

Parameters	Parameter	Description
	memberDef	Specify the memberDef object that defines the member for which you are implementing shared-nothing persistence.
	dataStore	Returns a string that indicates the directory path to the directory that holds the data store for shared-nothing persistence.

Remarks If you have implemented shared-nothing persistence, use the `tibasMemberDef_GetDataStore()` function to return the directory path for the directory where the persisted data store is stored.

See Also [tibasMemberDef_SetDataStore\(\)](#)

tibasMemberDef_SetRemoteDiscovery()

Function

Declaration `tibas_status` `tibasMemberDef_SetRemoteDiscovery`(
 `tibasMemberDef` `memberDef`,
 `const char*` `remote`)

Purpose Sets the discovery URL for a remote member.

Parameters	Parameter	Description
	<code>memberDef</code>	The <code>memberDef</code> object that identifies the remote member.
	<code>remote</code>	The discovery URL for the remote member.

Remarks Use the `tibasMemberDef_SetRemote()` function to specify the discovery URL for a remote space member.

See Also [tibasMemberDef_GetRemoteDiscovery\(\)](#)

tibasMemberDef_GetRemoteDiscovery()

Function

Declaration

```
tibas_status tibasMemberDef_GetRemoteDiscovery(  
    tibasMemberDef memberDef,  
    const char**    remoteDiscovery)
```

Purpose

Gets the discovery URL for a specified remote member.

Parameters

Parameter	Description
memberDef	The memberDef object that identifies the remote member.
remoteDiscovery	Returns the remote discovery URL for the specified member.

Remarks

Use the `tibasMemberDef_GetRemoteDiscovery()` function to return the name for a remote space member.

See Also

[tibasMemberDef_SetRemoteDiscovery\(\)](#)

tibasMemberDef_SetRemoteListen()

Function

Declaration `tibas_status` `tibasMemberDef_SetRemoteListen`(
 `tibasMemberDef` `memberDef`,
 `const char*` `remoteListen`

Purpose Specifies a listen URL for a specified remote member.

Parameters	Parameter	Description
	<code>memberDef</code>	The <code>memberDef</code> object that identifies the remote member.
	<code>remoteListen</code>	Specifies the remote listen URL to be set for the specified member.

Remarks Use the `tibasMemberDef_SetRemoteListen()` function to specify the remote listen URL for a remote space member.

See Also [tibasMemberDef_SetRemoteDiscovery\(\)](#)

tibasMemberDef_SetListen()

Function

Declaration `tibas_status` tibasMemberDef_SetListen
 (tibasMemberDef* memberDef,
 const char* listen)

Purpose Sets the listen attribute for a specified memberDef.

Parameters	Parameter	Description
	memberDef	The memberDef for which you want to set the listen attribute.
	listen	The listen attribute set by the function.

Remarks Use the `tibasMemberDef_SetListen()` function to specify the listen attribute for a specified memberDef. Specifically, the listen attribute specifies the interface and port that the member uses to listen for incoming connections from new members in the metaspace.

For information on the format for the listen attribute, see [Listen URL format on page 55](#) of the *TIBCO ActiveSpaces Developer's Guide*.

See Also [tibasMemberDef_Create\(\)](#), [tibasMemberDef_SetMemberName\(\)](#),
[tibasMemberDef_SetRemoteDiscovery\(\)](#), [tibasMemberDef_SetDiscovery\(\)](#),
[tibasMemberDef_GetListen\(\)](#), [tibasMemberDef_SetDiscovery\(\)](#)

tibasMemberDef_SetDiscovery()

Function

```
Declaration      tibas_status tibasMemberDef_SetDiscovery
                   (tibasMemberDef memberDef,
                   const char*      discovery)
```

Purpose Sets the discovery attribute of the specified MemberDef object.

Parameters

Parameter	Description
memberDef	The memberDef for which you want to set the discovery attribute.
discovery	The discovery attribute set by the function.

Remarks	Use the <code>tibasMemberDef_SetDiscovery()</code> function to set the discovery attribute for a specified <code>MemberDef</code> .
----------------	---

The discovery attribute specifies the discovery protocol and URL used to discover the other members of the metaspace on the network.

You can use one of the following discovery protocols:

- **PGM (Pragmatic General Multicast) Discovery** A reliable multicast protocol based on RFC 3208 that provides ordered or unordered, duplicate-free, multicast data.
- For information on the URL format for PGM, see [PGM \(Pragmatic General Multicast\) URL format on page 52](#) of the *TIBCO ActiveSpaces Developer's Guide*.



PGM is the recommended discovery method.

- **TIBCO Rendezvous Discovery** Discovery using TIBCO’s messaging system.
For information on the URL format for TIBCO Rendezvous discovery, see [TIBCO Rendezvous Discovery URL format on page 53](#) of the *TIBCO ActiveSpaces Developer’s Guide*.
- **TCP Discovery** When usage of multicast discovery is not desirable or possible, pure TCP discovery can be used. In this case, a number of metaspace members are designated as the “well known” members of the metaspace, and all metaspace members must specify this exact same list of well known members in their discovery URL

For information on the URL format for PGM, see [. on page 54](#) of the *TIBCO ActiveSpaces Developer's Guide*.



All members of a metaspace must use compatible discovery URLs. If two metaspace members attempt to connect to the same metaspace using different discovery protocols, two different metaspaces are created and the two members are in different metaspaces.

See Also [tibasMemberDef_Create\(\)](#), [tibasMemberDef_GetDiscovery\(\)](#),
[tibasMemberDef_SetListen\(\)](#), [tibasMemberDef_GetListen\(\)](#)

tibasMemberDef_GetListen()

Function

Declaration `tibas_status` `tibasMemberDef_GetListen`
 (`tibasMemberDef` `memberDef`,
 const `char**` `listen`)

Purpose Gets the listen attribute of the connection definition.

Parameters	Parameter	Description
	<code>memberDef</code>	The <code>memberDef</code> for which you want to return the listen attribute.
	<code>listen</code>	The listen attribute for the <code>memberDef</code> .

Remarks Use the `tibasMemberDef_GetListen()` function to return the listen attribute for a specified `memberDef`.

See Also [tibasMemberDef_Create\(\)](#), [tibasMemberDef_SetMemberName\(\)](#),
[tibasMemberDef_SetRemoteDiscovery\(\)](#), [tibasMemberDef_SetDiscovery\(\)](#),
[tibasMemberDef_GetListen\(\)](#), [tibasMemberDef_SetDiscovery\(\)](#)

tibasMemberDef_GetDiscovery()

Function

Declaration `tibas_status` tibasMemberDef_GetDiscovery
 (tibasMemberDef memberDef,
 const char** discovery)

Purpose Returns the discovery attribute for a specified memberDef.

Parameters	Parameter	Description
	memberDef	The memberDef for which you want to return the discovery attribute.
	discovery	The discovery attribute returned by the function.

Remarks Use the `tibasMemberDef_GetDiscovery()` function to return the discovery URL for a specified memberDef.

See Also [tibasMemberDef_SetDiscovery\(\)](#), [tibasMemberDef_SetListen\(\)](#),
 [tibasMemberDef_GetListen\(\)](#)

tibasMemberDef_SetContext()

Function

Declaration `tibas_status tibasMemberDef_SetContext(
 tibasMemberDef memberDef,
 tibasTuple context);`

Purpose Specifies a tuple that contains context information for a specified memberDef.

Parameters	Parameter	Description
	memberDef	Specify the memberDef for which to specify context information.
	context	A tuple that contains context information.

Remarks Use the `tibasMemberDef_SetContext()` function to specify a tuple that contains data that is unique for a specified member.

For example, you might want to specify the name of an application that the member is associated with. You can place this information in a tuple and include it in the memberDef for the member so that it can be referenced by your application.

See Also [tibasMember_GetContext\(\)](#)

tibasMemberDef_GetSecurityPolicyFile()

Function

Declaration `tibas_status tibasMemberDef_GetSecurityPolicyFile(
 tibasMemberDef memberDef,
 const char** policyFile)`

Purpose Returns the location and filename for the policy file that is in effect for a specified member.

Parameters	Parameter	Description
	memberDef	Specify the memberDef object that identifies the member for which you want to retrieve the security policy file.
	policyFile	Specify a pointer to a NULL-terminated string buffer that will hold the contents of the policy file that is returned.

Remarks Use the `tibasMemberDef_GetSecurityPolicyFile()` function to return the policy file that is in effect for a specified space member that is acting as a security domain controller.

 If the member is not functioning as a security domain controller then it is a security domain requestor and will not have a security policy file; instead it will have a security token file. You can use the [tibasMemberDef_GetSecurityTokenFile\(\)](#) function to return the location and filename for the security policy file.

See Also [tibasMemberDef_SetSecurityPolicyFile\(\)](#),
 [tibasMemberDef_GetSecurityTokenFile\(\)](#)

tibasMemberDef_GetSecurityTokenFile()

Function

Declaration `tibas_status` `tibasMemberDef_GetSecurityTokenFile`(
 `tibasMemberDef` `memberDef`,
 `const char**` `tokenFile`

Purpose Returns the directory path and filename for the security token file that is in effect for a specified member.

Parameters	Parameter	Description
	<code>memberDef</code>	Specify the <code>memberDef</code> object that identifies the member for which you want to retrieve the security token file.
	<code>tokenFile</code>	Specify a pointer to a NULL-terminated string buffer that will hold the security token file information that is returned.

Remarks Use the `tibasMemberDef_GetSecurityTokenFile()` function to return the directory path and filename for the security token file that is used by a specified member.

 If the member is acting as a security domain controller, then there is not a security token file associated with the member; instead, there is a security policy file. You can use the `tibasMemberDef_GetSecurityPolicyFile()` function to return information about the security policy file.

See Also `tibasMemberDef_SetSecurityTokenFile()`,
 `tibasMemberDef_GetSecurityPolicyFile()`

tibasMemberDef_GetAuthenticationCallback()

Function

Declaration `tibas_status` tibasMemberDef_GetAuthenticationCallback(
 tibasMemberDef memberDef,
 tibasMetaspace_AuthenticationCallback* callback)

Purpose Returns the authentication callback setting for a specified memberDef, if the callback function has been specified.

Parameters	Parameter	Description
	memberDef	Specify the memberDef object that identifies the member for which you want to return the callback function name.
	callback	Returns a pointer to the callback function, if one exists.

Remarks Use the `tibasMemberDef_GetAuthenticationCallback()` function to return a pointer to the authentication callback function that has been specified for a specified member, if a callback function has been specified.

See Also [tibasMemberDef_SetAuthenticationCallback\(\)](#)

tibasMemberDef_SetSecurityPolicyFile()

Function

Declaration `tibas_status` tibasMemberDef_SetSecurityPolicyFile(
 tibasMemberDef memberDef,
 const char* policyFile)

Purpose Sets the directory path and filename for the security policy file for a specified member.

Parameters	Parameter	Description
	memberDef	Specify the memberDef object that identifies the member for which you want to set the security policy file.
	policyFile	A string that specifies that directory path and filename for the security policy file. For example, "c:/tibco/as/2.1/lib/mypolicyfile.txt."

Remarks Use the `tibasMemberDef_SetSecurityPolicyFile()` function to specify the security policy file for a specified member that will function as a security domain controller.

If the member is a security domain requestor, a policy file is not required; instead you must specify a security token file. You can use the [tibasMemberDef_SetSecurityTokenFile\(\)](#) to specify the directory path and filename for the security token file.

See Also [tibasMemberDef_SetSecurityTokenFile\(\)](#),
 [tibasMemberDef_GetSecurityPolicyFile\(\)](#)

tibasMemberDef_SetSecurityTokenFile()

Function

Declaration `tibas_status` tibasMemberDef_SetSecurityTokenFile(
 tibasMemberDef memberDef,
 const char* tokenFile)

Purpose Specifies the directory path and security token filename for a specified member.

Parameters

Parameter	Description
memberDef	Specify the memberDef object that identifies the member for which you want to set the security token file.
tokenFile	A string that specifies that directory path and filename for the security token file. For example, c:/tibco/as/2.1/lib/mytokenfile.txt.

Remarks Use the `tibasMemberDef_SetSecurityTokenFile()` function to specify the security token file for a specified member that will function as a security domain requester.

If the member is a security domain controller, a security token file is not required; instead you must specify a security policy file. You can use the [tibasMemberDef_SetSecurityPolicyFile\(\)](#) to specify the directory path and filename for the security policy file.

See Also [tibasMemberDef_GetSecurityTokenFile\(\)](#),
 [tibasMemberDef_SetSecurityPolicyFile\(\)](#)

tibasMemberDef_SetAuthenticationCallback()

Function

Declaration `tibas_status` tibasMemberDef_SetAuthenticationCallback(
 tibasMemberDef memberDef,
 tibasMetaspace_AuthenticationCallback callback,
 void* closure)

Purpose Specifies a callback routine that allows you to customize how user authentication information is retrieved for users.

If you invoke tibasMemberDef_SetAuthenticationCallback() to specify a callback routine for your application and the security policy file for the domain that controls the metaspace which your application is joining specifies user password or X509v3 authentication, then the callback routine is called when users try to connect to the domain.

If there is no callback routine, ActiveSpaces provides a default mechanism, which prompts the user for the required information on the default console. In such a case, passwords are never echoed on the console.

Parameters

Parameter	Description
memberDef	Specify the memberDef object that identifies the member for which you want to set the authentication callback.
callback	Specifies the name of the callback function used to process authentication information
closure	A pointer to a closure value that is returned to your application when the authentication callback function completes. This can be a text string, or any information that your application needs to keep track of during the authentication process.

Remarks Use the tibasMemberDef_SetAuthenticationCallback() function to specify the name of a callback function that is used to retrieve authentication credentials for users requesting connection to a specified member.

The authentication callback routine must conform to the following function prototype, which is defined in the security.h header file:

```
tibas_status TIBAS_COMMON_API tibasMemberDef memberDef,  
tibasMetaspace_AuthenticationCallback callback, void* closure);
```

The authentication callback routine takes one argument—a `tibasAuthenticationInfo` struct that is defined in the `security.h` header file as follows:

```
typedef struct _tibasAuthenticationInfo
{
    tibas_authenticationMethod authenticationMethod;
    const char* metaspacename;
    tibasUserCredential* credential;
    const char* authHint;
} tibasAuthenticationInfo;
```

The `tibasAuthenticationInfo` structure includes the following members:

- **authenticationMethod** Based on the value assigned to the authentication setting in the security policy file that the security domain is using, ActiveSpaces assigns one of the following values to the `authenticationMethod` data type:
 - **AUTH_USERPWD** ActiveSpaces sets the authentication method to this value is set when the security policy file specifies `authentication=userpwd`.
 - **AUTH_X509V3** ActiveSpaces sets the authentication method to this value when the security policy file specifies `authentication=x509`.
- **metaspacename** The currently connected metaspacename is assigned.
- **credential** A pointer to a `tibasUserCredential` structure that is used to process the authentication information.
- **authHint** ActiveSpaces populates this element with the authorization hint that is provided in the security policy file that the active security domain is using the hint keyword, for example, `hint=SystemLoginInformation`.

Using the Credential Member

The credential member is a `tibasUserCredential` structure. This structure contains a set of members that the callback function uses to store authentication information that is passed to it, such as authentication domains, usernames, and passwords.

The `tibasUserCredential` is defined in the `security.h` header file. It has the following definition:

```
typedef struct _tibasUserCredential
{
    char domain[64];
    char username[64];
    char keyfile[256];
    char password[64];
}
```

```
} tibasUserCredential;
```

You use the members of the credential structure to pass to ActiveSpaces the authentication information provided by users requesting access to security domains. The elements are defined as follows:

- **domain** Passes authentication domain information to ActiveSpaces, if applicable (for example, for Windows systems)
- **username** Passes the username for authentication.
- **keyfile** If X509 authentication is used, passes either the account or keyfile password.
- **password** Passes the user password.

Example For a code example showing how to use an authentication callback function, see the reference article on the ASUserAuthentication example program in the *TIBCO ActiveSpaces Developers Guide* ([ASUserAuthenticator on page 186](#)).

See Also [tibasMemberDef_GetAuthenticationCallback\(\)](#)

tibasMemberDef_SetTimeout()

Function

Declaration `tibas_status tibasMemberDef_SetTimeout(
 tibasMemberDef memberDef,
 tibas_long* timeout)`

Purpose Specifies a timeout interval for a specified memberDef.

Parameters

Parameter	Description
memberDef	Specify the memberDef object for which you want to specify a timeout interval.
timeout	The timeout interval, in milliseconds.

Remarks Use the tibasMemberDef_SetTimeout() function to specify a timeout value for space operations on a specified space member.

The default timeout for a space operation is 60 seconds. After the timeout interval expires, a space operation times out and does not complete, unless it has already completed. You can use `tibasMemberDef_SetTimeout()` to change the default timeout value.

See Also [tibasMemberDef_Create\(\)](#)

Chapter 7 **FieldDef**

Field definitions describe the format of the data that will be stored in the space.

A valid space definition must contain at least one field definition. Field definitions are created by the factory object and can be put (or taken) from space definitions.

Field definitions can also be re-used and put into as many space definitions as needed (for example when using some fields as *foreign keys* to correlate tuples stored in different spaces).

Topics

- [FieldDef Operations](#)

FieldDef Operations

The following table lists the FieldDef operations :

Table 12 FieldDef

Function or Type	Description	Page
tibasFieldDef_Create()	Returns a new FieldDef instance. The default value of the field is “not nullable.”	223
tibasFieldDef_CreateEx()	Returns a new fieldDef instance and allows you to specify <code>fieldName</code> , <code>fieldType</code> , and key attributes.	225
tibasFieldDef_GetName()	Returns the field name for a specified fieldDef object.	226
tibasFieldDef_GetType()	Returns the type that is set for a specified fieldDef.	227
tibasFieldDef_IsNullable()	Indicates whether a specified field is nullable.	228
tibasFieldDef_SetNullable()	Sets a specified field as nullable or not nullable.	229
tibasFieldDef_SetEncrypted()	Specifies whether the data in a specified tuple field is to be encrypted.	230
tibasFieldDef_IsEncrypted()	Indicates whether a specified field is set as an encrypted field.	231
tibasFieldDef_Free()	Destroys allocated resources and frees memory.	232

tibasFieldDef_Create()

Function

Declaration `tibas_status` tibasFieldDef_Create
 (tibasFieldDef* fieldDef,
 char* fieldName,
 tibas_type fieldType)

Purpose Returns a new FieldDef instance. The default value of the field is “not nullable.”

Parameters

Parameter	Description
fieldDef	The new instance of a field definition returned by the function
fieldName	The field name.
fieldType	The field type.

Remarks Use the `tibasFieldDef_Create()` function to create a new field definition. The field definition is returned in a `fieldDef` object that you pass to the function.

The `fieldName` parameter specifies a string, enclosed in double quotes, that represents the field name.

The `fieldType` parameter specifies the type for the field. This can be any of the data types listed in the `tibas_type` enumeration (See [tibas_type](#), page 512 for the enumeration that lists the data types).

After you have created the fields for a tuple by calling `tibasFieldDef_Create()`, you can assign the fields to a `spaceDef` object by calling the [tibasSpaceDef_PutFieldDef\(\)](#) function and assign a key value by calling the [tibasSpaceDef_SetKey\(\)](#) function.

Example

```
tibasSpaceDef spaceDef = NULL;
tibasFieldDef keyField = NULL;
tibasFieldDef valueField = NULL;
tibasFieldDef timeField = NULL;
char const* spaceName = "myspace";

tibasFieldDef_Create(&keyField, "key", TIBAS_INTEGER);
tibasFieldDef_Create(&valueField, "value", TIBAS_STRING);
tibasFieldDef_SetNullable(valueField, TIBAS_TRUE);
tibasFieldDef_Create(&timeField, "time", TIBAS_DATETIME);
tibasFieldDef_SetNullable(timeField, TIBAS_TRUE);

tibasSpaceDef_Create(&spaceDef);
tibasSpaceDef_PutFieldDef(spaceDef, keyField);
tibasSpaceDef_PutFieldDef(spaceDef, valueField);
tibasSpaceDef_PutFieldDef(spaceDef, timeField);
```

```
tibasSpaceDef_SetKey(spaceDef, "key")
```

See Also [tibasFieldDef_SetNullable\(\)](#), [tibasSpaceDef_PutFieldDef\(\)](#),
[tibasSpaceDef_SetKey\(\)](#)

tibasFieldDef_CreateEx()

Function

Declaration

`tibas_status` `tibasFieldDef_CreateEx`
(`tibasFieldDef*` `fieldDef`,
`char*` `fieldName`,
`tibas_type` `fieldType`,
`tibas_boolean` `key`)

Purpose

Returns a new `fieldDef` instance and allows you to specify `fieldName`, `fieldType`, and `key` attributes.

Parameters

Parameter	Description
<code>fieldDef</code>	The new instance of a field definition returned by the function
<code>fieldName</code>	The field name.
<code>fieldType</code>	The field type.
<code>key</code>	A boolean value indicating whether the field is a key field or not.

Remarks

Use the `tibasFieldDef_CreateEx()` function to create a new `fieldDef` object and assign `fieldName`, `fieldType`, and `key` attributes.

See Also

[tibasFieldDef_Create\(\)](#)

tibasFieldDef_GetName()

Function

Declaration `tibas_status` `tibasFieldDef_GetName`
 (`tibasFieldDef` `fieldDef`,
 `char**` `fieldName`)

Purpose Returns the field name for a specified `fieldDef` object.

Parameters	Parameter	Description
	<code>fieldDef</code>	The <code>fieldDef</code> object for which you want to return a field name.
	<code>fieldName</code>	The name of the field returned by the function. You application must not modify or free the string, as it is owned by the <code>fieldDef</code> object.

Remarks Use the `tibasFieldDef_GetName()` function to return the field name for a specified `fieldDef` object.

See Also [tibasFieldDef_Create\(\)](#), [tibasFieldDef_CreateEx\(\)](#)

tibasFieldDef_GetType()

Function

Declaration `tibas_status` `tibasFieldDef_GetType`
 (`tibasFieldDef` `fieldDef`,
 `tibas_type*` `type`)

Purpose Returns the type that is set for a specified `fieldDef`.

Parameters	Parameter	Description
	<code>fieldDef</code>	The <code>fieldDef</code> object for which you want to return the field type.
	<code>type</code>	The type of the field returned by the function.

Remarks Use the `tibasFieldDef_GetType()` function to return the type that is set for a specified `fieldDef`.

See Also [tibasFieldDef_Create\(\)](#), [tibasFieldDef_CreateEx\(\)](#), [tibasFieldDef_GetName\(\)](#)

tibasFieldDef_IsNullable()

Function

```
Declaration      tibas_boolean tibasFieldDef_IsNullable
                   (tibasFieldDef fieldDef)
```

Purpose	Indicates whether a specified field is nullable. If the field is nullable, returns TIBAS_TRUE. If the field is not nullable, returns TIBAS_FALSE.
----------------	--

Parameters	Parameter	Description
	fieldDef	Specifies the FieldDef for the field whose nullable attribute is to be queried.

Remarks	Use the <code>tibasFieldDef_IsNullable()</code> function to determine whether a specified field is nullable. Fields marked as nullable do not need to be assigned values, but if a value is assigned, the data type must match or be able to be upcasted.
----------------	---

See Also [tibasFieldDef_SetNullable\(\)](#)

tibasFieldDef_SetNullable()

Function

Declaration `tibas_status` `tibasFieldDef_SetNullable`
 (`tibasFieldDef` `fieldDef`,
 `tibas_boolean` `nullable`)

Purpose Sets a specified field as nullable or not nullable.

Parameters	Parameter	Description
	<code>fieldDef</code>	A <code>fieldDef</code> object that identifies the field that you want to set as nullable or not nullable.
	<code>nullable</code>	Valid values are <code>TIBAS_TRUE</code> or <code>TIBAS_FALSE</code> .

Remarks Use the `tibasFieldDef_SetNullable()` function to set a specified field as nullable or not nullable. To set the field as nullable, specify `TIBAS_TRUE` for the nullable parameter; to set it as not nullable, specify `TIBAS_FALSE`.

Fields marked as nullable do not need to be assigned values, but if a value is assigned, the data type must match or be able to be upcasted.

See Also [tibasFieldDef_IsNullable\(\)](#)

tibasFieldDef_SetEncrypted()

Function

Declaration `tibas_status tibasFieldDef_SetEncrypted(
 tibasFieldDef fieldDef,
 tibas_boolean secured)`

Purpose Specifies whether the data in a specified tuple field is to be encrypted.

Parameters

Parameter	Description
fieldDef	Specify the fieldDef object for the field that you want to set as encrypted.
secured	<p>A tibas_boolean value that specifies whether the field data will be encrypted.</p> <p>To encrypt the field data, specify TIBAS_TRUE. If the data should be unencrypted, specify TIBAS_FALSE.</p>

Remarks Use the tibasFieldDef_SetEncrypted() function to specify that the data for a specified field definition is encrypted.

When data is put into a field that is defined to be encrypted, the data is encrypted while it resides in memory in the data grid and when it is persisted with shared-nothing persistence.

Certain types of fields in a space should not be encrypted. Do not encrypt fields that are used:

- As keys or indexes
- In filters for searching through the data in a space

See Also [tibasFieldDef_IsEncrypted\(\)](#),

tibasFieldDef_IsEncrypted()

Function

Declaration	<code>tibas_boolean</code> <code>tibasFieldDef_IsEncrypted(tibasFieldDef fieldDef)</code>	
Purpose	Indicates whether a specified field is set as an encrypted field.	
Parameters	Parameter	Description
	<code>fieldDef</code>	Specify the <code>fieldDef</code> for the field for which you want to determine the encryption setting.
Remarks	<p>Use the <code>tibasFieldDef_IsEncrypted()</code> function to query whether a specified field is encrypted.</p> <p>If the field is an encrypted field, returns <code>TIBAS_TRUE</code>; otherwise, returns <code>TIBAS_FALSE</code>.</p>	
See Also	tibasFieldDef_SetEncrypted()	

tibasFieldDef_Free()

Function

```
Declaration      tibas_status tibasFieldDef_Free
                   (tibasFieldDef* fieldDef)
```

Purpose	Destroys allocated resources and frees memory.
----------------	--

Parameters

Parameter	Description
fieldDef	A fieldDef object that identifies the field definition that you want to free.

Remarks Use the `tibasFieldDef_Free()` function to free the allocated resources and memory used by a field definition.

See Also [tibasFieldDef_Create\(\)](#)

Chapter 8 **FieldDefList**

This chapter explains the `FieldDefList` operations. Field definition lists are holders for collections of `FieldDef` objects.

Topics

- [FieldDefList Operations, page 234](#)

FieldDefList Operations

The following table lists the FieldDefList operations:

Table 13 *FieldDefList*

Function or Type	Description	Page
tbasFieldDefList_Size()	Returns the number of FieldDef objects in a specified FieldDefList.	235
tbasFieldDefList_Get()	Returns the FieldDef object at a specified index position in a FieldDefList.	236
tbasFieldDefList_Free()	Frees a specified FieldDefList.	237

tibasFieldDefList_Size()

Function

Declaration `tibas_status tibasFieldDefList_Size(
 tibasFieldDefList fieldDefList,
 tibas_int* size)`

Purpose Returns the number of FieldDef objects in a specified FieldDefList.

Parameters	Parameter	Description
	fieldDefList	The TIBCO ActiveSpaces entity on which the function is invoked.
	size	The number of FieldDef objects in the specified fieldDefList.

Remarks Use the `tibasFieldDefList_Size()` function to return the number of fieldDef objects in a specified fieldDefList.

Before you can call this function, you must create one or more fieldDefList objects and populate a field definition list.

See Also [tibasFieldDefList_Get\(\)](#)

tibasFieldDefList_Get()

Function

Declaration `tibas_status tibasFieldDefList_Get(
 tibasFieldDefList fieldDefList,
 tibasFieldDef* fieldDef,
 tibas_int index)`

Purpose Returns the FieldDef object at a specified index position in a FieldDefList.

Parameters	Parameter	Description
	fieldDefList	The TIBCO ActiveSpaces entity on which the function is invoked.
	fieldDef	Returns the FieldDef definition.
	index	The index position for which the FieldDef object is to be returned.

Remarks Use the `tibasFieldDefList_Get()` function to return the fieldDef at a specified index position in a specified fieldDefList.

Before you can call this function, you must create one or more fieldDefList objects and populate a field definition list. Also, you should call the [tibasFieldDefList_Size\(\)](#) function to determine the number of elements in the fieldDefList. This will enable you to avoid specifying an index position that is beyond the upper limit of the field definition list.

See Also [tibasFieldDefList_Size\(\)](#)

tibasFieldDefList_Free()

Function

Declaration `tibas_status` tibasFieldDefList_Free(
 tibasFieldDefList* fieldDefList)

Purpose Frees a specified FieldDefList.

Parameters	Parameter	Description
	fieldDefList	Specifies the fieldDefList that you want to free.

Remarks Use the tibasFieldDefList_Free() function to free the resources and memory used for a specified FieldDefList.

See Also [tibasFieldDefList_Get\(\)](#)

Chapter 9 **KeyDef**

This chapter explains the KeyDef operations. KeyDef functions allow you to set up a primary key for searches within a space, and set the type of indexing used with the key.

Topics

- [KeyDef Operations](#)

KeyDef Operations

The following table lists the KeyDef operations:

Table 14 *KeyDef*

Function or Type	Description	Page
tibasKeyDef_Create()	Creates a primary key and specifies the fields for the key.	241
tibasKeyDef_GetIndexType()	Returns the index type for a specified primary key.	242
tibasKeyDef_SetIndexType()	Sets the index type used with a specified primary key.	243
tibasKeyDef_GetFieldNames()	Returns a list of the field names associated with a specified primary key.	244
tibasKeyDef_SetFieldNames()	Sets the field names used as the primary key in a KeyDef.	245
tibasKeyDef_Free()	Frees the memory used for a specified primary key definition.	246

tibasKeyDef_Create()

Function

Declaration `tibas_status tibasKeyDef_Create(
 tibasKeyDef* keyDef,
 char* fieldNames)`

Purpose Creates a primary key and specifies the fields for the key.

Parameters	Parameter	Description
	keyDef	Specifies the name of the key definition for the primary key that you are creating.
	fieldNames	A comma-separated list that specifies the field names to be used as the primary key.

Remarks Use the `tibasKeyDef_Create()` function to create a primary key and specify the fields for the key..

The `fieldnames` parameter specifies a comma separated list of field names to be used as the primary key.

To activate the key definition:

- If you do not already have a `spaceDef` object to which to assign the key definition, use the `tibasSpaceDef_Create()` function to a `spaceDef` object.
- Call the `tibasSpaceDef_SetKeyDef()` function to assign the `KeyDef` to a specified `SpaceDef` so that it can be used as the primary key for searches within the space.



When you call `tibasKeyDef()` to create a key, the default index type is `TIBAS_INDEX_HASH`. You can change the index type by calling the `tibasKeyDef_SetIndexType()` function.

Note also that if you need to change the primary key, you must free the space by calling the `tibasSpace_Free()` function and then reconnect to the space using the new key definition.

See Also `tibasKeyDef_SetIndexType()`, `tibasKeyDef_SetFieldNames()`

tibasKeyDef_GetIndexType()

Function

Declaration `tibas_status tibasKeyDef_GetIndexType(
 tibasKeyDef keyDef,
 tibas_indexType* indexType)`

Purpose Returns the index type for a specified primary key.

Parameters	Parameter	Description
	keyDef	Specifies the key definition for the key whose index type is to be retrieved.
	tibas_indexType	Returns the index type currently set for the key.

Remarks Use the `tibasKeyDef_GetIndexType()` function to return the index type for a specified primary key.

The `tibas_indexType` parameter returns the index type, which can be one of the following:

- **TIBAS_INDEX_HASH** Specifies that indexing is done by using a hash table.
- **TIBAS_INDEX_TREE** Specifies that indexing is done using a tree.

See Also [tibasKeyDef_SetIndexType\(\)](#)

tibasKeyDef_SetIndexType()

Function

Declaration `tibas_status` `tibasKeyDef_SetIndexType`(
 `tibasKeyDef` `keyDef`,
 `tibas_indexType` `indexType`)

Purpose Sets the index type used with a specified primary key.

Parameters	Parameter	Description
	<code>keyDef</code>	Specifies the KeyDef for the key whose index type you want to set.
	<code>tibas_indexType</code>	Specifies the index type to set.

Remarks Use the `tibasKeyDef_SetIndex()` function to specify the index type for a specified KeyDef. The index type can be one of the following:

- **TIBAS_INDEX_HASH** Specifies that indexing is done by using a hash table.
- **TIBAS_INDEX_TREE** Specifies that indexing is done using a tree.

When you initially create an index using the `tibasKeyDef_Create()` function, the index type is set to the default value, `TIBAS_INDEX_HASH`, which specifies that indexing is done by using a hash table.

You can call `tibasKeyDef_SetIndexType()` to set it to the alternate value `TIBAS_INDEX_TREE`, which specifies that indexing is done using a tree. Or, if it has been previously set to `TIBAS_INDEX_TREE`, you can set it back to `TIBAS_INDEX_HASH`.

See Also [tibasKeyDef_GetIndexType\(\)](#)

tibasKeyDef_GetFieldNames()

Function

Declaration `tibas_status tibasKeyDef_GetFieldNames(
 tibasKeyDef keyDef,
 tibasStringList* fieldNames)`

Purpose Returns a list of the field names associated with a specified primary key.

Parameters

Parameter	Description
keyDef	Specifies the KeyDef for the key whose key values you want to return.
fieldNames	Pointer to a comma-separated list of the key values currently set for the specified primary key.

Remarks Use the `tibasKeyDef_GetFieldNames()` function to return the field names associated with a specified primary key.

See Also [tibasKeyDef_SetFieldNames\(\)](#), [tibasSpaceDef_SetKeyDef\(\)](#),
 [tibasSpaceDef_GetKeyDef\(\)](#)

tibasKeyDef_SetFieldNames()

Function

Declaration `tibas_status tibasKeyDef_SetFieldNames(
 tibasKeyDef keyDef,
 char* fieldNames)`

Purpose Sets the field names used as the primary key in a KeyDef.

Parameters	Parameter	Description
	keyDef	Specifies the KeyDef for the key whose key definition you want to set.
	fieldNames	Pointer to a comma-separated list of the key values to set for the specified primary key.

Remarks Use the `tibasKeyDef_SetFieldNames()` function to specify the key names that are used as a primary key for a specified KeyDef.

 Use the `fieldNames` parameter to specify a comma-separated list of values that indicates the field names to use as the primary key.

See Also [tibasKeyDef_Create\(\)](#), [tibasKeyDef_GetFieldNames\(\)](#)

tibasKeyDef_Free()

Function

Declaration `tibas_status tibasKeyDef_Free(
 tibasKeyDef* keyDef)`

Purpose Frees the memory used for a specified primary key definition.

Parameters	Parameter	Description
	keyDef	Specifies the KeyDef for the primary key definition to free.

Remarks Use the `tibasKeyDef_Free()` function to free the memory associated with a specified KeyDef.

See Also [tibasKeyDef_Create\(\)](#)

Chapter 10 IndexDef

This chapter explains the `IndexDef` operations. The `IndexDef` operations allow you to create secondary indexes, set and retrieve the fields in the indexes, and specify the type of index that is used.

Secondary indexes provide additional search capabilities for browsers, and are useful in querying data that is stored in the persistent database that is created if you implement shared-all persistence.

For additional information on the use of keys and indexes in TIBCO ActiveSpaces data grids, see the “Key Fields” and “Indexes” sections in chapter 3 of the *TIBCO ActiveSpaces Developer’s Guide*, “Performing Basic ActiveSpaces Tasks.”

Topics

- [IndexDef Operations](#)

IndexDef Operations

The following table lists the IndexDef operations:

Table 15 *IndexDef*

Function or Type	Description	Page
tibasIndexDef_Create()	Creates a secondary index definition.	249
tibasIndexDef_GetName()	Returns the index name for a specified IndexDef.	250
tibasIndexDef_SetName()	Specifies a new name for a specified index.	251
tibasIndexDef_GetIndexType()	Returns the index type that is set for a specified index.	252
tibasIndexDef_SetIndexType()	Sets the index type for a specified index.	253
tibasIndexDef_GetFieldNames()	Retrieves the field names currently set for a specified secondary index.	254
tibasIndexDef_SetFieldNames()	Sets the field names used for a specified secondary index.	255
tibasIndexDef_Free()	Frees the memory used by a specified secondary index and deletes the index.	256
tibasIndexDefList_Size()	Returns the number of elements in a specified list of IndexDefs.	257
tibasIndexDefList_Get()	Returns the Index definition for an element at a specified position in an IndexDefList for a specified IndexDef element.	258
tibasIndexDefList_Free()	Frees the memory used by a specified IndexDefList and deletes the definition list.	259

tibasIndexDef_Create()

Function

Declaration `tibas_status tibasIndexDef_Create(
 tibasIndexDef* indexDef,
 char* indexName,
 char* fieldNames)`

Purpose Creates a secondary index definition.

Parameters

Parameter	Description
indexDef	Pointer to an IndexDef that is created.
indexName	Specifies the name of the secondary index.
fieldNames	Pointer to a list of comma-separated values that specifies the fields to be used for the secondary index.

Remarks Use the `tibasIndexDef_Create()` function to create a secondary index.

Secondary indexes provide a way to search for data that supplements the primary index that you create with the [tibasKeyDef_Create\(\)](#) function. Indexes are built when spaces are created (or loaded) and use memory to help locate matching tuples faster than it takes to iterate through a long record. Secondary indexes can speed up searches significantly.

By default, `tibasIndexDefCreate()` assigns the index type for the index the value `TIBAS_INDEX_HASH`. You can use the [tibasIndexDef_SetIndexType\(\)](#) function to change the index to `TIBAS_INDEX_TREE` if you want to use a tree index.

After you create the index by calling `tibasIndexDef_Create()`, you must add it to the `spaceDef` for the space by calling the [tibasSpaceDef_AddIndexDef\(\)](#) function. Then, when you join a space, you pass the `spaceDef` to the [tibasMetaspace_DefineSpace\(\)](#) function.

See Also [tibasIndexDef_SetName\(\)](#), [tibasIndexDef_SetIndexType\(\)](#),
[tibasIndexDef_SetFieldNames\(\)](#), [tibasIndexDef_Free\(\)](#),
[tibasSpaceDef_AddIndexDef\(\)](#), [tibasMetaspace_DefineSpace\(\)](#)

tibasIndexDef_GetName()

Function

Declaration `tibas_status` `tibasIndexDef_GetName(`
 `tibasIndexDef` `indexDef,`
 `char**` `indexName)`

Purpose Returns the index name for a specified IndexDef.

Parameters

Parameter	Description
<code>indexDef</code>	Specifies the IndexDef for the index for which you want to get the index name.
<code>indexName</code>	Returns the name of the index.

Remarks Use the `tibasIndexDef_GetName()` function to return the index name for a specified IndexDef.

See Also [tibasIndexDef_SetName\(\)](#)

tibasIndexDef_SetName()

Function

Declaration

```
tibas_status tibasIndexDef_SetName(  
    tibasIndexDef indexDef,  
    char*          indexName)
```

Purpose

Specifies a new name for a specified index.

Parameters

Parameter	Description
indexDef	The IndexDef for the index for which you want to specify a name.
indexName	Specify a string indicating the index name.

Remarks

Use the `tibasIndexDef_SetName()` function to assign a name to a specified index.

See Also

[tibasIndexDef_GetName\(\)](#)

tibasIndexDef_GetIndexType()

Function

Declaration `tibas_status` `tibasIndexDef_GetIndexType`(
 `tibasIndexDef` `indexDef`,
 `tibas_indexType*` `indexType`)

Purpose Returns the index type that is set for a specified index.

Parameters	Parameter	Description
	<code>indexDef</code>	The IndexDef for the index for which you want to return the index type.
	<code>indexType</code>	Returns the index type for the index.

Remarks Use the `tibasIndexDef_GetIndexType()` function to return the index type for a specified index. The index type can be one of the following:

- **TIBAS_INDEX_HASH** Specifies that indexing is done by using a hash table.
- **TIBAS_INDEX_TREE** Specifies that indexing is done using a tree.

See Also [tibasIndexDef_SetIndexType\(\)](#)

tibasIndexDef_SetIndexType()

Function

Declaration `tibas_status` `tibasIndexDef_SetIndexType`(
 `tibasIndexDef` `indexDef`,
 `tibas_indexType` `indexType`)

Purpose Sets the index type for a specified index.

Parameters	Parameter	Description
	<code>indexDef</code>	Specifies the IndexDef for the index for which you want to set the index type.
	<code>indexType</code>	Specifies the index type.

Remarks Use the `tibasIndexDef_SetIndexType()` function to return the index type for a specified index. The index type can be one of the following:

- **TIBAS_INDEX_HASH** Specifies that indexing is done by using a hash table.
- **TIBAS_INDEX_TREE** Specifies that indexing is done using a tree.

When you first create an index definition by calling the [tibasIndexDef_Create\(\)](#) function, the index type is set to the default value—TIBAS_INDEX_HASH. If you want to change the index type to TIBAS_INDEX_TREE, you can use `tibasIndexDefSetIndexType()` to change it.

See Also [tibasIndexDef_GetIndexType\(\)](#)

tibasIndexDef_GetFieldNames()

Function

Declaration `tibas_status tibasIndexDef_GetFieldNames(
 tibasIndexDef indexDef,
 tibasStringList* fieldNames)`

Purpose Retrieves the field names currently set for a specified secondary index.

Parameters	Parameter	Description
	indexDef	Specifies the IndexDef for the index for which you want to retrieve field names.
	fieldNames	Pointer to a list of comma-separated values indicating the field names defined for the index.

Remarks Use the `tibasIndexDef_GetFieldNames()` function to retrieve the field names that are currently set for a specified secondary index.

See Also [tibasIndexDef_SetFieldNames\(\)](#)

tibasIndexDef_SetFieldNames()

Function

Declaration

`tibas_status` `tibasIndexDef_SetFieldNames(`
 `tibasIndexDef` `indexDef,`
 `char*` `fieldNames)`

Purpose

Sets the field names used for a specified secondary index.

Parameters

Parameter	Description
<code>indexDef</code>	Specifies the IndexDef for the index for which you want to set field names.
<code>fieldNames</code>	Pointer to a list of comma-separated values that specify the field names to assign to the index.

Remarks

Use the `tibasIndexDef_SetFieldNames()` function to change the field names that are currently set for a specified secondary index.

When you create an index definition by calling the `tibasIndexDef_Create()` function, you define the fields used for the secondary index. You can call the `tibasIndexDef_SetFieldNames()` function to change the field names.

See Also

`tibasIndexDef_Create()`, `tibasIndexDef_GetFieldNames()`

tibasIndexDef_Free()

Function

Declaration `tibas_status tibasIndexDef_Free(
 tibasIndexDef* indexDef)`

Purpose Frees the memory used by a specified secondary index and deletes the index.

Parameters	Parameter	Description
	indexDef	Specifies the IndexDef for the index that you want to delete.

Remarks Use the `tibasIndexDef_Free()` function to free the memory used by a specified secondary index and free the index.

See Also [tibasIndexDef_Create\(\)](#)

tibasIndexDefList_Size()

Function

Declaration `tibas_status` `tibasIndexDefList_Size`(
 `tibasIndexDefList` `indexDefList`,
 `tibas_int*` `size`)

Purpose Returns the number of elements in a specified list of IndexDefs.

Parameters	Parameter	Description
	<code>indexDefList</code>	A list of comma-separated values that specifies the indexes.
	<code>size</code>	Returns the number of index elements in the IndexDefList.

Remarks Use the `tibasIndexDefList_Size()` function to return the number of elements in a specified list of IndexDefs.

Use the size value returned by `tibasIndexDefList_Size()` as to determine the upper limit for the index value used with the `tibasIndexDefList_Get()` function.

See Also `tibasIndexDefList_Get()`

tibasIndexDefList_Get()

Function

Declaration `tibas_status` `tibasIndexDefList_Get`(
 `tibasIndexDefList` `indexDefList`,
 `tibasIndexDef*` `indexDef`,
 `tibas_int` `index`)

Purpose Returns the Index definition for an element at a specified position in an IndexDefList for a specified IndexDef element.

Parameters	Parameter	Description
	<code>indexDefList</code>	Specify the IndexDefList to search.
	<code>indexDef</code>	Returns the IndexDef at the specified index position.
	<code>index</code>	Specify an index value. If the index value you specify is above the maximum number of entries in the list, the function returns an error.

Remarks Use the `tibasIndexDefList_Get()` function to return the `indexDef` that is at a specified index position in an `indexDefList`.

 The value returned by the `tibasIndexDefList_Size()` function provides the number of entries in the list.

See Also `tibasIndexDefList_Size()`

tibasIndexDefList_Free()

Function

Declaration	<code>tibas_status</code> <code>tibasIndexDefList_Free(tibasIndexDefList* indexDefList)</code>	
Purpose	Frees the memory used by a specified IndexDefList and deletes the definition list.	
Parameters	Parameter	Description
	<code>indexDefList</code>	Specifies the name of the IndexDefList to free.
Remarks	Use the <code>tibasIndexDefList_Free()</code> function to free the memory used by a specified IndexDefList and delete the definition list.	
See Also	<code>tibasIndexDefList_Size()</code> , <code>tibasIndexDefList_Get()</code>	

Chapter 11 **BrowserDef**

This chapter explains the BrowserDef operations. Browsers are used to iterate through tuples stored in a space. A browser can have either of two scopes, time scope and distribution scope, which are defined by setting the values of fields in the browser's BrowserDef object.

Topics

- [BrowserDef Operations](#)

BrowserDef Operations

The following table lists the BrowserDef operations:

Table 16 *BrowserDef*

Function or Type	Description	Page
tibasBrowserDef_Create()	Creates a new instance of a BrowserDef.	263
tibasBrowserDef_CreateEx()	Returns a new BrowserDef instance and allows you to set the attributes of the browserDef.	264
tibasBrowserDef_GetDistributionScope()	Returns the distribution scope of the browser.	266
tibasBrowserDef_GetTimeout()	Returns the timeout value associated with a specified browserDef.	267
tibasBrowserDef_GetTimeScope()	Returns the time scope associated with a specified browserDef.	268
tibasBrowserDef_SetDistributionScope()	Sets the distribution scope associated with a specified browserDef (browser).	269
tibasBrowserDef_SetTimeScope()	Sets the time scope associated with a specified browserDef (browser).	270
tibasBrowserDef_SetTimeout()	Sets the timeout value for a specified browserDef (browser).	271
tibasBrowserDef_Free()	Destroys allocated resources and frees memory.	272

tibasBrowserDef_CreateEx()

Function

Declaration `tibas_status` `tibasBrowserDef_CreateEx`
 (`tibasBrowserDef*` `browserDef`,
 `tibas_int` `timeout`,
 `tibas_timeScope` `timeScope`,
 `tibas_distributionScope` `distributionScope`)

Purpose Returns a new BrowserDef instance and allows you to set the attributes of the browserDef.

Parameter	Description
browserDef	The new instance of the browser definition object returned by the function.
timeout	The time in milliseconds or TIBAS_WAIT_FOREVER or TIBAS_NO_WAIT.
timeScope	Specifies the time scope for the browser. Possible values are: <ul style="list-style-type: none">TIBAS_TIME_SCOPE_ALLTIBAS_TIME_SCOPE_SNAPSHOT (the default value)TIBAS_TIME_SCOPE_NEWTIBAS_TIME_SCOPE_NEW_EVENTS
distributionScope	The distribution scope can be one of the following: <ul style="list-style-type: none">TIBAS_DISTRIBUTION_SCOPE_ALL (the default)TIBAS_DISTRIBUTION_SCOPE_SEEDED

Remarks The `tibasBrowserDef_CreateEx()` function is an extended version of the [tibasBrowserDef_Create\(\)](#) function. Use `tibasBrowserDef_CreateEx()` to create a browser definition and directly set attributes of the browserDef.

The timeout is the amount of time for which a space browser's `tibasBrowser_Next()` function can block while waiting for something new in the space to execute a `tibasBrowser_Next()` on. If there is still nothing new for the browser to perform a `next()` operation on at the end of the timeout, the `tibasBrowser_Next()` function returns null.

You can specify one of the following with the timeout parameter:

- The timeout value in milliseconds.

- **TIBAS_WAIT_FOREVER** Specifies that the browser will not time out.
- **TIBAS_NO_WAIT** Specifies that the browser will not wait before executing another `tibasBrowser_Next()` operation.



The browser's timeout value is ignored when the browser's time scope is set to `TIBAS_TIME_SCOPE_SNAPSHOT`. In this case, any invocation of the `tibasBrowser_Next()` function on the browser once all of the tuples in the snapshot have been iterated through immediately returns `NULL`.

The `timescope` parameter specifies the time scope for the browser, and can have one of one of the following values:

- **TIBAS_TIME_SCOPE_ALL** Start the browser with all the tuples currently in the space and update it continuously updated when tuples are added.
- **TIBAS_TIME_SCOPE_SNAPSHOT** Start the browser with all the tuples in the space at the time the browser is created (or initial values), but do not update it when new tuples are added. This is the default value.
- **TIBAS_TIME_SCOPE_NEW** Start the browser with no tuples, but update it with new tuples that are created in the space.
- **TIBAS_TIME_SCOPE_NEW_EVENTS** This value is only used with event browsers. The browser starts empty and is updated when new events occur.

The distribution scope can be one of the following:

- **TIBAS_DISTRIBUTION_SCOPE_ALL** The browser browses over all the tuples (or associated events) in the space.
- **TIBAS_DISTRIBUTION_SCOPE_SEEDED** The browser browses over only the tuples (or associated events) that have been distributed to the member creating the browser.

After you create the `browserDef`, you pass the `browserDef` to the [tibasMetaspace_Browse\(\)](#) function or the [tibasSpace_Browse\(\)](#) function to enable the browser.

See Also [tibasBrowserDef_Create\(\)](#), [tibasMetaspace_Browse\(\)](#), [tibasSpace_Browse\(\)](#)

tibasBrowserDef_GetDistributionScope()

Function

[illegible]

Purpose	Returns the distribution scope of the browser.
----------------	--

Parameters

Parameter	Description
browserDef	The TIBCO ActiveSpaces entity on which the function is invoked.
distributionScope	Returns the distribution scope for the browser. The distribution scope can be one of the following: <ul style="list-style-type: none">TIBAS_DISTRIBUTION_SCOPE_ALLTIBAS_DISTRIBUTION_SCOPE_SEEDED

Remarks	Use the <code>tibasBrowserDef_GetDistributionScope()</code> function to return the distribution scope for a specified <code>browserDef</code> . The distribution scope can be one of the following:
----------------	---

- **TIBAS_DISTRIBUTION_SCOPE_ALL** The browser browses over all the tuples (or associated events) in the space.
- **TIBAS_DISTRIBUTION_SCOPE_SEEDED** The browser browses over only the tuples (or associated events) that have been distributed to the member creating the browser.

See Also [tibasBrowserDef_Create\(\)](#), [tibasBrowserDef_CreateEx\(\)](#),

tibasBrowserDef_GetTimeout()

Function

Declaration `tibas_status` tibasBrowserDef_GetTimeout(
 tibasBrowserDef browserDef,
 tibas_long* timeout)

Purpose Returns the timeout value associated with a specified browserDef.

Parameters	Parameter	Description
	browserDef	The TIBCO ActiveSpaces entity on which the function is invoked.
	timeout	The timeout in milliseconds.

Remarks Use the `tibasBrowserDef_GetTimeout()` function to return the timeout value currently specified for a specified browserDef.

See Also `tibasBrowserDef_Create()`, `tibasBrowserDef_SetTimeout()`

tibasBrowserDef_GetTimeScope()

Function

Declaration `tibas_status` tibasBrowserDef_GetTimeScope(
 tibasBrowserDef browserDef,
 tibas_timeScope* timeScope)

Purpose Returns the time scope associated with a specified browserDef.

Parameters	Parameter	Description
	browserDef	The TIBCO ActiveSpaces entity on which the function is invoked.
	timeScope	The time scope of the browser.

Remarks Use the `tibasBrowserDef_GetTimeScope()` function to return the time scope value currently specified for a specified browserDef.

See Also `tibasBrowserDef_Create()`, `tibasBrowserDef_SetTimeScope()`

tibasBrowserDef_SetDistributionScope()

Function

[illegible]

Purpose Sets the distribution scope associated with a specified browserDef (browser).

Parameters

Parameter	Description
browserDef	The TIBCO ActiveSpaces entity on which the function is invoked.
distributionScope	<p>The distribution scope can be one of the following:</p> <ul style="list-style-type: none"> TIBAS_DISTRIBUTION_SCOPE_ALL (the default) TIBAS_DISTRIBUTION_SCOPE_SEEDED

Remarks	Use the <code>tibasBrowserDef_SetDistributionScope()</code> function to specify the distribution scope for a specified <code>browserDef</code> object and its associated browser.
----------------	---

The distribution scope can be one of the following:

- **TIBAS_DISTRIBUTION_SCOPE_ALL** The browser browses over all the tuples (or associated events) in the space.
- **TIBAS_DISTRIBUTION_SCOPE_SEEDED** The browser browses over only the tuples (or associated events) that have been distributed to the member creating the browser.

See Also [tibasBrowserDef_Create\(\)](#), [tibasBrowserDef_CreateEx\(\)](#), [tibasBrowserDef_GetDistributionScope\(\)](#)

tibasBrowserDef_SetTimeScope()

Function

```
Declaration      tibas_status tibasBrowserDef_SetTimeScope
                   (tibasBrowserDef browserDef,
                    int_timescope      timescope)
```

Purpose Sets the time scope associated with a specified browserDef (browser).

Parameters

Parameter	Description
browserDef	The TIBCO ActiveSpaces entity on which the function is invoked.
timescope	<p>Specifies the time scope for the browser. Possible values are:</p> <ul style="list-style-type: none">• TIBAS_TIME_SCOPE_ALL• TIBAS_TIME_SCOPE_SNAPSHOT (the default value)• TIBAS_TIME_SCOPE_NEW• TIBAS_TIME_SCOPE_NEW_EVENTS

Remarks	Use the <code>tibasBrowserDef_SetTimeScope()</code> function to set the time scope associated with a specified browserDef and its associated browser.
----------------	---

The `timescope` parameter specifies the time scope for the browser, and can have one of one of the following values:

- **TIBAS_TIME_SCOPE_ALL** Start the browser with all the tuples currently in the space and update it continuously updated when tuples are added.
- **TIBAS_TIME_SCOPE_SNAPSHOT** Start the browser with all the tuples in the space at the time the browser is created (or initial values), but do not update it when new tuples are added. This is the default value.
- **TIBAS_TIME_SCOPE_NEW** Start the browser with no tuples, but update it with new tuples that are created in the space.
- **TIBAS_TIME_SCOPE_NEW_EVENTS** This value is only used with event browsers. The browser starts empty and is updated when new events occur.

See Also [tibasBrowserDef_Create\(\)](#), [tibasBrowserDef_CreateEx\(\)](#), [tibasBrowserDef_GetTimeScope\(\)](#)

tibasBrowserDef_SetTimeout()

Function

Declaration `tibas_status tibasBrowserDef_SetTimeout
(tibasBrowserDef browserDef,
tibas_long timeout)`

Purpose Sets the timeout value for a specified browserDef (browser).

Parameters

Parameter	Description
browserDef	The TIBCO ActiveSpaces entity on which the function is invoked.
timeout	The time in milliseconds or TIBAS_WAIT_FOREVER or TIBAS_NO_WAIT.

Remarks Use the `tibasBrowserDef_SetTimeout()` function to specify a timeout value for a specified browserDef (browser).

The timeout is the amount of time for which a space browser's `tibasBrowser_Next()` function can block while waiting for something new in the space to execute a `tibasBrowser_Next()` on. If there is still nothing new for the browser to perform a `next()` operation on at the end of the timeout, the `tibasBrowser_Next()` function returns null.

You can specify one of the following with the timeout parameter:

- The timeout value in milliseconds.
- **TIBAS_WAIT_FOREVER** Specifies that the browser will not time out.
- **TIBAS_NO_WAIT** Specifies that the browser will not wait before executing another `tibasBrowser_Next()` operation.



The browser's timeout value is ignored when the browser's time scope is set to `TIBAS_TIME_SCOPE_SNAPSHOT`. In this case, any invocation of the `tibasBrowser_Next()` function on the browser once all of the tuples in the snapshot have been iterated through immediately returns NULL.

See Also `tibasBrowserDef_Create()`, `tibasBrowserDef_CreateEx()`,
`tibasBrowserDef_GetTimeout()`

tibasBrowserDef_Free()

Function

```
Declaration      tibas_status tibasBrowserDef_Free
                   (tibasBrowserDef* browserDef)
```

Purpose	Destroys allocated resources and frees memory.
----------------	--

Parameters

Parameter	Description
browserDef	The TIBCO ActiveSpaces entity on which the function is invoked.

Remarks	Use the <code>tibasBrowserDef_Free()</code> function to free the resources and memory associated with a specified <code>browserDef</code> .
----------------	---

See Also [tibasBrowserDef_Create\(\)](#)

Chapter 12 **Browser**

This chapter explains Browser operations in TIBCO ActiveSpaces.

Topics

- [Browser Operations](#)

Browser Operations

The following table lists the Browser operations :

Table 17 Space

Function or Type	Description	Page
Browser Functions		
tibasBrowser_Next()	Returns the next tuple being browsed, or NULL if the browser timed out waiting for a new tuple to be put into the space.	275
tibasBrowser_Free()	Destroys allocated resources and frees memory.	276

tibasBrowser_Next()

Function

Declaration	<code>tibas_status</code> <code>tibasBrowser_Next</code> (<code>tibasBrowser</code> browser, tibastuple* value)							
Purpose	Returns the next tuple being browsed, or NULL if the browser timed out waiting for a new tuple to be put into the space.							
Parameters	<table><tr><th>Parameter</th><th>Description</th></tr><tr><td>browser</td><td>The browser on which the function is invoked.</td></tr><tr><td>value</td><td>Returns the value for the next tuple that is browsed.</td></tr></table>		Parameter	Description	browser	The browser on which the function is invoked.	value	Returns the value for the next tuple that is browsed.
Parameter	Description							
browser	The browser on which the function is invoked.							
value	Returns the value for the next tuple that is browsed.							
Remarks	<p>Use the <code>tibasBrowser_Next()</code> function to get the next tuple in a space for which you have implemented a space browser.</p> <p>If the browser times out while waiting for a new tuple to be put into the space, the function returns NULL.</p>							
See Also	tibasBrowserDef_GetTimeout() , tibasBrowserDef_SetTimeout()							

tibasBrowser_Free()

Function

```
Declaration      tibas_status tibasBrowser_Free
                   (tibasBrowser* browser)
```

Purpose	Destroys allocated resources and frees memory.
----------------	--

Parameters

Parameter	Description
browser	The browser on which the function is invoked.

Remarks	Use the <code>tibasBrowser_Free()</code> function to free the resources used by a specified space browser.
----------------	--

The browser parameter specifies the browser object that was returned when the browser was activated by calling the `tibasMetaspace_Browse()` function or the `tibasSpace_Browse()` function.

See Also [tibasMetaspace_Browse\(\)](#), [tibasSpace_Browse\(\)](#)

Chapter 13 **EventBrowserDef**

This chapter explains the `EventBrowserDef` operations. Event browsers are used to iterate through the stream of events that occur in a space. An event browser can have either of two scopes: time scope and distribution scope, which are defined by setting the values of fields in the event browser's `EventBrowserDef` object.

Topics

- [EventBrowserDef Operations](#)

EventBrowserDef Operations

The following table lists the EventBrowserDef operations:

Table 18 BrowserDef

Function or Type	Description	Page
tibasEventBrowserDef_Create()	Creates a new instance of an EventBrowserDef.	279
tibasEventBrowserDef_CreateEx()	Returns a new EventBrowserDef instance.	280
tibasEventBrowserDef_GetDistributionScope()	Returns the distribution scope for a specified event browser definition (eventBrowserDef).	282
tibasEventBrowserDef_GetTimeout()	Gets the timeout value associated with the event browser definition.	283
tibasEventBrowserDef_GetTimeScope()	Gets the time scope associated with a specified eventBrowserDef.	284
tibasEventBrowserDef_SetDistributionScope()	Sets the distribution scope associated with the event browser.	285
tibasEventBrowserDef_SetTimeScope()	Sets the time scope associated with the event browser.	286
tibasEventBrowserDef_SetTimeout()	Sets the timeout value for a specified eventBrowserDef.	287
tibasEventBrowserDef_Free()	Destroys the allocated resources and frees the memory associated with a specified eventBrowserDef.	288

tibasEventBrowserDef_Create()

Function

Declaration	<code>tibas_status</code> <code>tibasEventBrowserDef_Create</code> (<code>tibasEventBrowserDef*</code> <code>eventBrowserDef</code>)				
Purpose	Creates a new instance of an EventBrowserDef.				
Parameters	<table><tr><th>Parameter</th><th>Description</th></tr><tr><td><code>eventBrowserDef</code></td><td>The new instance of an EventBrowserDef returned by the function.</td></tr></table>	Parameter	Description	<code>eventBrowserDef</code>	The new instance of an EventBrowserDef returned by the function.
Parameter	Description				
<code>eventBrowserDef</code>	The new instance of an EventBrowserDef returned by the function.				
Remarks	<p>Use the <code>tibasEventBrowserDef_Create()</code> function to create an EventBrowserDef object. The browserDef defines the characteristics of an event browser, which allows your application to browse through the tuples in a space based on the browser characteristics defined in the browserDef.</p> <p>The space browser is created when you pass the eventBrowserDef to the <code>tibasMetaspace_BrowseEvents()</code> function or the <code>tibasSpace_BrowseEvents()</code> function.</p> <p>If you create the event browser definition by calling <code>tibasEventBrowserDef_Create()</code>, then you must set the attributes of the browser by calling additional <code>tibasEventBrowserDef</code> functions. Call:</p> <ul style="list-style-type: none">• <code>tibasEventBrowserDef_SetTimeout()</code> to set the a timeout value for the event browser.• <code>tibasEventBrowserDef_SetTimeScope()</code> to set a time scope for the event browser.• <code>tibasEventBrowserDef_SetDistributionScope()</code> to set a distribution scope. <p>You can both create the browser and directly set the browser attributes by calling the <code>tibasEventBrowserDef_CreateEx()</code> function.</p>				
See Also	<code>tibasEventBrowserDef_SetTimeout()</code> , <code>tibasEventBrowserDef_SetTimeScope()</code> , <code>tibasEventBrowserDef_SetDistributionScope()</code>				

tibasEventBrowserDef_CreateEx()

Function

Declaration `tibas_status` tibasEventBrowserDef_CreateEx
 (`tibasEventBrowserDef*` `eventBrowserDef`,
 `tibas_int` `timeout`,
 `tibas_timeScope` `timeScope`,
 `tibas_distributionScope` `distributionScope`)

Purpose Returns a new EventBrowserDef instance.

Parameter	Description
<code>eventBrowserDef</code>	The new instance of the event browser definition object returned by the function.
<code>timeout</code>	The value of the timeout to be set in the returned EventBrowserDef.
<code>timeScope</code>	<p>The time scope values for the event browser.</p> <p>The time scope can be one of the following:</p> <ul style="list-style-type: none">• <code>TIBAS_TIME_SCOPE_SNAPSHOT</code>• <code>TIBAS_TIME_SCOPE_NEW</code>• <code>TIBAS_TIME_SCOPE_NEW_EVENTS</code>• <code>TIBAS_TIME_SCOPE_ALL</code> (the default).
<code>distributionScope</code>	The distribution scope to be set in the returned EventBrowserDef.

Remarks Use the `tibasEventBrowserDef_CreateEx()` function to create an `eventBrowserDef` object that defines the attributes of an event browser and directly set the attributes.

The timeout is the amount of time for which an event browser's `tibasEventBrowser_Next()` function can block while waiting for something new in the space to execute a `tibasEventBrowser_Next()` on. If there is still nothing new for the browser to perform a `next()` operation on at the end of the timeout, the `tibasEventBrowser_Next()` function returns null.

You can specify one of the following with the `timeout` parameter:

- The timeout value in milliseconds.

- **TIBAS_NO_WAIT** Specifies that the browser will not wait before executing another `tibasEventBrowser_Next()` operation.



The browser's timeout value is ignored when the browser's time scope is set to `TIBAS_TIME_SCOPE_SNAPSHOT`. In this case, any invocation of the `tibasBrowser_Next()` function on the browser once all of the tuples in the snapshot have been iterated through immediately returns `NULL`.

The `timescope` parameter specifies the time scope for the event browser, and can have one of the following values:

- **TIBAS_TIME_SCOPE_ALL** The event browser starts with all the events currently in the space at creation time (which will be presented as an initial set of PUT events) and then is continuously updated according to changes in the space.
- **TIBAS_TIME_SCOPE_SNAPSHOT** The event browser contains only PUT events corresponding to the tuples stored in the space at creation time.
- **TIBAS_TIME_SCOPE_NEW** The event browser starts empty and is updated only with events related to new or updated tuples in the space.
- **TIBAS_TIME_SCOPE_NEW_EVENTS** The event browser starts empty and is updated with all events that occur in the space after creation time.

The distribution scope can be:

- **TIBAS_DISTRIBUTION_SCOPE_ALL** (default setting) The event browser browses events related to all tuples in the space.
- **TIBAS_DISTRIBUTION_SCOPE_SEEDED** The event browser browses only events associated with the tuples in the space that are seeded by this member.

After you create the `eventBrowserDef`, you pass it to the `tibasMetaspace_BrowseEvents()` function or the `tibasSpace_BrowseEvents()` function to enable the browser.

See Also [tibasEventBrowserDef_Create\(\)](#), [tibasMetaspace_BrowseEvents\(\)](#), [tibasSpace_BrowseEvents\(\)](#)

tibasEventBrowserDef_GetDistributionScope()

Function

Declaration `tibas_status` tibasEventBrowserDef_GetDistributionScope
 (`tibasEventBrowserDef` `eventBrowserDef`,
 `tibas_distributionScope*` `distributionScope`)

Purpose Returns the distribution scope for a specified event browser definition
 (`eventBrowserDef`).

Parameters	Parameter	Description
	<code>eventBrowserDef</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>distributionScope</code>	Possible values are <code>TIBAS_DISTRIBUTION_SCOPE_ALL</code> or <code>TIBAS_DISTRIBUTION_SCOPE_SEEDED</code> .

Remarks Use the `tibasEventBrowserDef_GetDistributionScope()` function to return the distribution scope currently set in a specified `eventBrowserDef`.

The distribution scope can be:

- **TIBAS_DISTRIBUTION_SCOPE_ALL** (default setting) The event browser browses events related to all tuples in the space.
- **TIBAS_DISTRIBUTION_SCOPE_SEEDED** The event browser browses only events associated with the tuples in the space that are seeded by this member.

See Also [tibasEventBrowserDef_Create\(\)](#), [tibasEventBrowserDef_SetDistributionScope\(\)](#)

tibasEventBrowserDef_GetTimeScope()

Function

Declaration `tibas_status` `tibasEventBrowserDef_GetTimeScope`
 (`tibasEventBrowserDef` `eventBrowserDef`,
 `tibas_timeScope*` `timeScope`)

Purpose Gets the time scope associated with a specified `eventBrowserDef`.

Parameters	Parameter	Description
	<code>eventBrowserDef</code>	Specifies the <code>eventBrowserDef</code> for which you want to return the time scope. TIBCO ActiveSpaces entity on which the function is invoked.
	<code>timeScope</code>	The time scope value for the of the event browser.

Remarks Use the `tibasEventBrowserDef_GetTimeScope()` function to return the time scope setting currently associated with a specified `eventBrowserDef`.

The time scope value can be one of the following:

- **TIBAS_TIME_SCOPE_ALL** The event browser starts with all the events currently in the space at creation time (which will be presented as an initial set of PUT events) and then is continuously updated according to changes in the space.
- **TIBAS_TIME_SCOPE_SNAPSHOT** The event browser contains only PUT events corresponding to the tuples stored in the space at creation time.
- **TIBAS_TIME_SCOPE_NEW** The event browser starts empty and is updated only with events related to new or updated tuples in the space.
- **TIBAS_TIME_SCOPE_NEW_EVENTS** The event browser starts empty and is updated with all events that occur in the space after creation time.

See Also [tibasEventBrowserDef_Create\(\)](#)[tibasEventBrowserDef_SetTimeScope\(\)](#)

tibasEventBrowserDef_SetDistributionScope()

Function

Declaration

tibas_status

(tibasEventBrowserDef

tibas_distributionScope*

tibasEventBrowserDef_SetDistributionScope

eventBrowserDef,

distributionScope)

Purpose

Sets the distribution scope associated with the event browser.

Parameters

Parameter	Description
eventBrowserDef	The TIBCO ActiveSpaces entity on which the function is invoked.
distributionScope	The distribution scope can be either TIBAS_DISTRIBUTION_SCOPE_ALL (the default) or TIBAS_DISTRIBUTION_SCOPE_SEEDED.

Remarks

Use the `tibasEventBrowserDef_SetDistributionScope()` function to specify the distribution scope associated with a specified `eventBrowserDef`.

The distribution scope can be:

- TIBAS_DISTRIBUTION_SCOPE_ALL** (default setting) The event browser browses events related to all tuples in the space.
- TIBAS_DISTRIBUTION_SCOPE_SEEDED** The event browser browses only events associated with the tuples in the space that are seeded by this member.

See Also

[tibasEventBrowserDef_Create\(\)](#), [tibasEventBrowserDef_GetDistributionScope\(\)](#)

tibasEventBrowserDef_SetTimeScope()

Function

Declaration `tibas_status` `tibasEventBrowserDef_SetTimeScope`
 (`tibasEventBrowserDef` `eventBrowserDef`,
 `int_timescope` `timescope`)

Purpose Sets the time scope associated with the event browser.

Parameters	Parameter	Description
	<code>eventBrowserDef</code>	Specifies the <code>eventBrowserDef</code> for which you want to return a time scope value.
	<code>timescope</code>	The time scope value for the event browser. The time scope can be one of the following: <ul style="list-style-type: none">• <code>TIBAS_TIME_SCOPE_SNAPSHOT</code>• <code>TIBAS_TIME_SCOPE_NEW</code>• <code>TIBAS_TIME_SCOPE_NEW_EVENTS</code>• <code>TIBAS_TIME_SCOPE_ALL</code> (the default).

Remarks Use the `tibasEventBrowserDef_SetTimeScope()` function to set the time scope associated with a specified `eventBrowserDef`.

The time scope value can be one of the following:

- **TIBAS_TIME_SCOPE_ALL** The event browser starts with all the events currently in the space at creation time (which will be presented as an initial set of PUT events) and then is continuously updated according to changes in the space.
- **TIBAS_TIME_SCOPE_SNAPSHOT** The event browser contains only PUT events corresponding to the tuples stored in the space at creation time.
- **TIBAS_TIME_SCOPE_NEW** The event browser starts empty and is updated only with events related to new or updated tuples in the space.
- **TIBAS_TIME_SCOPE_NEW_EVENTS** The event browser starts empty and is updated with all events that occur in the space after creation time.

See Also [tibasEventBrowserDef_Create\(\)](#), [tibasEventBrowserDef_GetTimeScope\(\)](#)

tibasEventBrowserDef_Free()

Function

Declaration	<code>tibas_status</code> <code>tibasEventBrowserDef_Free</code> (<code>tibasEventBrowserDef*</code> <code>eventBrowserDef</code>)					
Purpose	Destroys the allocated resources and frees the memory associated with a specified <code>eventBrowserDef</code> .					
Parameters	<table><tr><th>Parameter</th><th>Description</th></tr><tr><td><code>eventBrowserDef</code></td><td>The <code>eventBrowserDef</code> that you want to free.</td></tr></table>		Parameter	Description	<code>eventBrowserDef</code>	The <code>eventBrowserDef</code> that you want to free.
Parameter	Description					
<code>eventBrowserDef</code>	The <code>eventBrowserDef</code> that you want to free.					
Remarks	Use the <code>tibasEventBrowserDef_Free()</code> function to free the resources and memory used by a specified <code>eventBrowserDef</code> .					
See Also	<code>tibasEventBrowserDef_Create()</code>					

Chapter 14 **EventBrowser**

This chapter explains the eventBrowser operations.

Topics

- [EventBrowser Operations](#)

EventBrowser Operations

The following table lists the EventBrowser operations:

Table 19 EventBrowser

Function or Type	Description	Page
tibasEventBrowser_Next()	Returns the next event for the space that is being browsed.	292
tibasEventBrowser_Free()	Destroys allocated resources and frees memory.	291

tibasEventBrowser_Next()

Function

Declaration `tibas_status tibasEventBrowser_Next(
 tibasEventBrowser eventBrowser,
 tibasSpaceEvent* spaceEvent)`

Purpose Returns the next event for the space that is being browsed.

Parameters

Parameter	Description
eventBrowser	Specifies the event browser for which to return the next event.
spaceEvent	The event returned by the function.

Remarks Use the `tibasEventBrowser_Next()` function to get the next event for a space for which you have implemented an event browser.

 If the event browser times out while waiting for a new event, the function returns NULL.

See Also [tibasSpace_Browse\(\)](#), [tibasMetaspace_Browse\(\)](#)

Chapter 15 **ListenerDef**

This chapter explains the ListenerDef operations.

Topics

- [ListenerDef Operations](#)

ListenerDef Operations

The following table lists the ListenerDef operations:

Table 20 *ListenerDef*

Function or Type	Description	Page
tibasListenerDef_Create()	Returns a new ListenerDef instance.	295
tibasListenerDef_CreateEx()	Returns a new ListenerDef instance specifying additional parameters.	296
tibasListenerDef_GetTimeScope()	Returns the time scope attribute for a specified listenerDef.	298
tibasListenerDef_GetDistributionScope()	Returns the distribution scope for a specified listenerDef.	299
tibasListenerDef_SetTimeScope()	Sets the time scope associated with the listener.	300
tibasListenerDef_SetDistributionScope()	Sets the distribution scope for a listener.	300
tibasListenerDef_Free()	Destroys allocated resources and frees memory.	302

tibasListenerDef_Create()

Function

```
Declaration      tibas_status tibasListenerDef_Create
                   (tibasListenerDef*  listenerDef)
```

Purpose Returns a new ListenerDef instance.

Parameters

Parameter	Description
listenerDef	The new instance of ListenerDef returned by the function.

Remarks Use the `tibasListenerDefCreate()` function to create a `ListenerDef` object that you can use to define an event listener.

After you have defined the `ListenerDef` object, use the `tibasListenerDef_SetTimeScope()` and `tibasListenerDef_SetDistributionScope()` functions to specify the time scope and distribution scope for the listener.

See Also [tibasListenerDef_CreateEx\(\)](#), [tibasListenerDef_SetTimeScope\(\)](#), [tibasListenerDef_SetDistributionScope\(\)](#), [tibasListenerDef_Free\(\)](#)

tibasListenerDef_CreateEx()

Function

Declaration `tibas_status` `tibasListenerDef_CreateEx`
 (`tibasListenerDef*` `listenerDef`,
 `tibas_timeScope` `timeScope`,
 `tibas_distributionScope` `distributionScope`)

Purpose Returns a new ListenerDef instance specifying additional parameters.

Parameters	Parameter	Description
	<code>listenerDef</code>	The new instance of ListenerDef returned by the function.
	<code>timeScope</code>	Specifies the time scope for the ListenerDef. The time scope can be <code>TIBAS_TIME_SCOPE_SNAPSHOT</code> , <code>TIBAS_TIME_SCOPE_NEW</code> , <code>TIBAS_TIME_SCOPE_NEW_EVENTS</code> , or <code>TIBAS_TIME_SCOPE_ALL</code> .
	<code>distributionScope</code>	The distribution scope to be set in the returned ListenerDef.

Remarks Use the `tibasListenerDef_CreateEx()` function to create a `listenerDef` object and directly set the attributes for the `listenerDef`.

To activate the listener, call the `tibasMetaspace_Listen()` function or the `tibasSpace_Listen()` and specify the `listenerDef` object that you have created with `tibasListenerDef_Create()` or `tibasListenerDefCreateEx()`.

The `timeScope` parameter specifies the time scope, which can be one of the following:

- **TIBAS_TIME_SCOPE_ALL** The listener starts with all the tuples currently in the space at creation time (which will be presented as an initial set of PUT events) and then is continuously updated according to changes in the space.
- **TIBAS_TIME_SCOPE_SNAPSHOT** The listener contains only PUT events corresponding to the tuples stored in the space at creation time.
- **TIBAS_TIME_SCOPE_NEW** The listener starts empty and is updated only with events related to new or updated tuples in the space.
- **TIBAS_TIME_SCOPE_NEW_EVENTS** (the default setting) The listener starts empty and is updated with all events that occur in the space after creation time.

The `distributionScope` parameter specifies the distribution scope for the listener, which can be one of the following:

- **TIBAS_DISTRIBUTION_SCOPE_ALL** (default setting) The listener listens to events related to all tuples in the space.
- **TIBAS_DISTRIBUTION_SCOPE_SEEDED** The listener listens only to events associated with the tuples in the space that are seeded by this member.

See Also [tibasListenerDef_Create\(\)](#), [tibasMetaspace_Connect\(\)](#), [tibasSpace_Listen\(\)](#)

tibasListenerDef_GetTimeScope()

Function

Declaration `tibas_status tibasListenerDef_GetTimeScope`
 `(tibasListenerDef listenerDef,`
 `tibas_timeScope* timeScope)`

Purpose Returns the time scope attribute for a specified listenerDef.

Parameters	Parameter	Description
	listenerDef	The listenerDef for which you want to return a time scope.
	timeScope	The time scope can be TIBAS_TIME_SCOPE_SNAPSHOT, TIBAS_TIME_SCOPE_NEW, TIBAS_TIME_SCOPE_NEW_EVENTS, or TIBAS_TIME_SCOPE_ALL.

Remarks Use the `tibasListenerDef_GetTimeScope()` function to return the time scope attribute for a specified listenerDef.

The time scope value can be:

- **TIBAS_TIME_SCOPE_ALL** The listener starts with all the tuples currently in the space at creation time (which will be presented as an initial set of PUT events) and then is continuously updated according to changes in the space.
- **TIBAS_TIME_SCOPE_SNAPSHOT** The listener contains only PUT events corresponding to the tuples stored in the space at creation time.
- **TIBAS_TIME_SCOPE_NEW** The listener starts empty and is updated only with events related to new or updated tuples in the space.
- **TIBAS_TIME_SCOPE_NEW_EVENTS** The listener starts empty and is updated with all events that occur in the space after creation time.

See Also [tibasListenerDef_Create\(\)](#), [tibasListenerDef_SetTimeScope\(\)](#)

tibasListenerDef_GetDistributionScope()

Function

Declaration `tibas_status` `tibasListenerDef_GetDistributionScope`
 (`tibasListenerDef` `listenerDef`,
 `tibas_distributionScope*` `distributionScope`)

Purpose Returns the distribution scope for a specified `listenerDef`.

Parameters	Parameter	Description
	<code>listenerDef</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>distributionScope</code>	The distribution for the specified <code>listenerDef</code> . Can be either <code>TIBAS_DISTRIBUTION_SCOPE_ALL</code> or <code>TIBAS_DISTRIBUTION_SCOPE_SEEDED</code> .

Remarks Use the `tibasListenerDef_GetDistributionScope()` function to return the distribution scope for a specified `listenerDef`.

The distribution scope can be:

- **TIBAS_DISTRIBUTION_SCOPE_ALL** (default setting) The listener listens to events related to all tuples in the space.
- **TIBAS_DISTRIBUTION_SCOPE_SEEDED** The listener listens only to events associated with the tuples in the space that are seeded by this member.

See Also [tibasListenerDef_Create\(\)](#), [tibasListenerDef_SetDistributionScope\(\)](#)

tibasListenerDef_SetTimeScope()

Function

Declaration `tibas_status tibasListenerDef_SetTimeScope`
 `(tibasListenerDef listenerDef,`
 `tibas_timeScope timeScope)`

Purpose Sets the time scope associated with the listener.

Parameters

Parameter	Description
listenerDef	The TIBCO ActiveSpaces entity on which the function is invoked.
timeScope	<p>The time scope values for the listener.</p> <p>The time scope can be one of the following:</p> <ul style="list-style-type: none">• TIBAS_TIME_SCOPE_SNAPSHOT• TIBAS_TIME_SCOPE_NEW• TIBAS_TIME_SCOPE_NEW_EVENTS• TIBAS_TIME_SCOPE_ALL (the default).

Remarks Use the `tibasListenerDef_SetTimeScope()` function to set the time scope for events that a listener will listen for. You can specify the following values:

- **TIBAS_TIME_SCOPE_ALL** The listener starts with all the tuples currently in the space at creation time (which will be presented as an initial set of PUT events) and then is continuously updated according to changes in the space.
- **TIBAS_TIME_SCOPE_SNAPSHOT** The listener contains only PUT events corresponding to the tuples stored in the space at creation time.
- **TIBAS_TIME_SCOPE_NEW** The listener starts empty and is updated only with events related to new or updated tuples in the space.
- **TIBAS_TIME_SCOPE_NEW_EVENTS** The listener starts empty and is updated with all events that occur in the space after creation time.

After you set the attributes for the ListenerDef object, you can pass it to either the `tibasSpace_Listen()` function or the `tibasMetaspace_Listen()` function to attach the listener to a space.

See Also [tibasListenerDef_Create\(\)](#), [tibasListenerDef_CreateEx\(\)](#),
[tibasListenerDef_SetDistributionScope\(\)](#), [tibasSpace_Listen\(\)](#),
[tibasMetaspace_Listen\(\)](#)

tibasListenerDef_Free()

Function

```
Declaration      tibas_status tibasListenerDef_Free
                   (tibasListenerDef* listenerDef)
```

Purpose	Destroys allocated resources and frees memory.
----------------	--

Parameters

Parameter	Description
listenerDef	The TIBCO ActiveSpaces entity on which the function is invoked.

Remarks	Use the <code>tibasListenerDef_Free()</code> function to free the resources used by a specified <code>listenerDef</code> object.
----------------	--

See Also [tibasListenerDef_Create\(\)](#)

Chapter 16 **Listener**

This chapter explains the Listener operations.

Topics

- [Listener Operations, page 304](#)

Listener Operations

The following table lists the Listener operations:

Table 21 *Listener*

Function or Type	Description	Page
tibasListener_Create()	Creates a listener object.	305
tibasSpaceMemberListener_Create()	Creates a space member listener.	307
tibasMemberListener_Create()	Creates a member event listener.	308
tibasSpaceRemoteMemberListener_Create()	Creates a remote space member listener.	309
tibasRemoteMemberListener_Create()	Creates a remote member listener.	310
tibasSpaceStateListener_Create()	Creates a space state listener.	311
tibasSpaceDefListener_Create()	Creates a space definition listener.	312

tibasListener_Create()

Function

Declaration `tibas_status tibasListener_Create(
 tibasListener* listener,
 tibas_onEvent callback,
 void* closure)`

Purpose Creates a listener object and associates a specified callback function with it.

Parameters

Parameter	Description
listener	The new listener object that is returned.
callback	The name of the callback function.
closure	Pointer to closure data that will be passed to the callback function when the listener is invoked.

Remarks Use the `tibasListenerCreate()` function to create a listener object and associate a callback function with it.

The callback function that you specify with the `callback` parameter is called when you attach the listener to a metaspace on which you want to listen for events by calling the `tibasMetaspace_Listen()` function or the `tibasSpace_Listen()` function.

When you call `tibasListenerCreate()`, a listener object is created. the listener is then activated using the `tibasMetaspace_Listen()` or `tibasSpace_Listen()` functions. The callback function is passed a `tibasSpaceEvent` object whose type can be determined by invoking the `tibasSpaceEvent_GetType` function.

The callback function that you provide must be able to process all types of space events generated by ActiveSpaces:

- `TIBAS_EVENT_PUT` when a tuple is inserted, overwritten, or updated.
- `TIBAS_EVENT_TAKE` when a tuple is taken or removed.
- `TIBAS_EVENT_EXPIRE` when a tuple reaches the end of its time to live and expires from the space
- `TIBAS_EVENT_SEED` are when there is redistribution after a seeder joins or leaves, and the local node is seeding or unseeding. This is only applicable if the listener distribution scope is `SEEDED`.
- `TIBAS_EVENT_UNSEED` when there is redistribution after a seeder joins or leaves, and the local node is seeding or unseeding. This is only applicable if the listener distribution scope is `SEEDED`.

You can also specify that a current snapshot of the entries stored in the space (sometimes referred to as *initial values*) is prepended to the stream of events. In this case, the initial values of all the tuples contained in the space at the listener's creation time are seen as space events of type `PUT` preceding the current stream of events.

The `tibasListener` object returned by `tibasListener_Create()` is required input for the [tibasMetaspace_Listen\(\)](#) function.

The closure argument is a pointer to closure data that is passed to the callback function when the listener is invoked. If you do not want to pass closure data to the callback function, specify `NULL` for this argument.

See Also [tibasMetaspace_Listen\(\)](#), [tibasSpace_Listen\(\)](#),

tibasSpaceMemberListener_Create()

Function

Declaration

```
tibas_status tibasSpaceMemberListener_Create(  
    tibasListener* listener,  
    tibas_onSpaceMemberEvent callback,  
    void* closure)
```

Purpose

Creates a space member listener.

Parameters

Parameter	Description
listener	The new listener object that is returned.
callback	The name of the callback function.
closure	Pointer to closure data that will be passed to the callback function when the listener is invoked.

Remarks

Use the `tibasSpaceMemberListener_Create()` function to create a listener that listens for space member events.

See Also

[tibasSpaceRemoteMemberListener_Create\(\)](#),
[tibasRemoteMemberListener_Create\(\)](#)

tibasMemberListener_Create()

Function

Declaration `tibas_status tibasMemberListener_Create(
 tibasListener* listener,
 tibas_onMemberEvent callback,
 void* closure)`

Purpose Creates a member event listener.

Parameters	Parameter	Description
	listener	The new listener object that is returned.
	callback	The name of the callback function.
	closure	Pointer to closure data that will be passed to the callback function when the listener is invoked.

Remarks Use the `tibasMemberListener_Create()` function to create a listener that listens for metaspace member events.

See Also [tibasSpaceRemoteMemberListener_Create\(\)](#)

tibasSpaceRemoteMemberListener_Create()

Function

Declaration `tibas_status` tibasSpaceRemoteMemberListener_Create(
 tibasListener* listener,
 tibas_onSpaceRemoteMemberEvent callback,
 void* closure)

Purpose Creates a remote space member listener.

Parameters	Parameter	Description
	listener	The new listener object that is returned.
	callback	The name of the callback function.
	closure	Pointer to closure data that will be passed to the callback function when the listener is invoked.

Remarks Use the `tibasSpaceRemoteMemberListener_Create()` function to create a listener that listens for space member events from remote space members.

See Also [tibasSpaceMemberListener_Create\(\)](#), [tibasRemoteMemberListener_Create\(\)](#)

tibasRemoteMemberListener_Create()

Function

Declaration `tibas_status` `tibasRemoteMemberListener_Create`(
 `tibasListener*` `listener`,
 `tibas_onRemoteMemberEvent` `callback`,
 `void*` `closure`)

Purpose Creates a remote member listener.

Parameters	Parameter	Description
	<code>listener</code>	The new listener object that is returned.
	<code>callback</code>	The name of the callback function.
	<code>closure</code>	Pointer to closure data that will be passed to the callback function when the listener is invoked.

Remarks Use the `tibasRemoteMemberListener_Create()` function to create a listener that listens for remote member events.

See Also [tibasSpaceRemoteMemberListener_Create\(\)](#)

tibasSpaceStateListener_Create()

Function

Declaration

```
tibas_status tibasSpaceStateListener_Create(  
    tibasListener* listener,  
    tibas_onSpaceStateChange callback,  
    void* closure)
```

Purpose

Creates a space state listener.

Parameters

Parameter	Description
listener	The new listener object that is returned.
callback	The name of the callback function.
closure	Pointer to closure data that will be passed to the callback function when the listener is invoked.

Remarks

Use the `tibasSpaceStateListener_Create()` function to create a space state listener.

See Also

[tibasSpaceDefListener_Create\(\)](#)

tibasSpaceDefListener_Create()

Function

Declaration `tibas_status tibasSpaceDefListener_Create(
 tibasListener* listener,
 tibas_onSpaceDefDefine callback,
 void* closure)`

Purpose Creates a space definition listener.

Parameters	Parameter	Description
	listener	The new listener object that is returned.
	callback	The name of the callback function.
	closure	Pointer to closure data that will be passed to the callback function when the listener is invoked.

Remarks Use the `tibasSpaceDefListener_Create()` function to create a listener that listens for changes to space definition.

See Also [tibasSpaceStateListener_Create\(\)](#)

Chapter 17

SpaceEvent, SpaceMemberEvent, SpaceRemoteMemberEvent, and RemoteMemberEvent

This chapter explains the various types of event operations:

Typically, ActiveSpaces applications use the SpaceEvent functions in a callback function that processes events from an event browser.

The other types of event operations are performed on events that are returned to listeners:

- **SpaceMemberEvent Operations** Space member events are generated when a member joins or leaves a space or changes its role from seeder to leech or from leech to seeder. These events are returned to space member listeners.
- **MemberEvent Operations** Member events are generated when a member joins or leaves a metaspace or changes its role from member to manager. These events are returned to member event listeners.
- **RemoteMemberEvent Operations** Remote member event operations are generated when a remote member joins or leaves a metaspace. These Events are returned to remote member listeners.
- **RemoteSpaceMemberEvent Operations** Remote space member events are generated when a remote member joins or leaves a space. These events are returned to remote space member listeners.

Topics

- [Space Event Operations](#)

Space Event Operations

The following table lists the operations in alphabetical order:

Table 22 *SpaceEvent*

Function or Type	Description	Page
Space Event Operations		
tibasSpaceEvent_Free()	Destroys allocated resources and frees memory.	319
tibasSpaceEvent_GetTuple()	Returns the tuple that generated a specified space event.	316
tibasSpaceEvent_GetSpace()	Returns a Space object that identifies the space from which a space event was generated.	318
tibasSpaceEvent_GetType()	Returns the type of an event generated by ActiveSpaces.	317
tibasSpaceEvent_HasOldTuple()	Checks whether a specified event has an existing tuple associated with it.	320
tibasSpaceEvent_GetOldTuple()	For a put event, returns the tuple associated with the event and updates the hash table for the space.	321
tibasSpaceEvent_GetMetaspaceName()	Returns the metaspace name that is associated with a specified event.	322
tibasSpaceEvent_GetSpaceName()	Returns the space name associated with a specified space event.	323
Space Member Operations		
tibasSpaceMemberEvent_GetType()	Returns the event type for a specified member event.	324
tibasSpaceMemberEvent_GetSpaceName()	Returns the space name for a specified space member event.	325

Table 22 *SpaceEvent*

Function or Type	Description	Page
tibasSpaceMemberEvent_GetDistributionRole()	Returns the distribution role for a specified space member event.	326
tibasSpaceMemberEvent_GetMember()	Returns the member ID for a specified space member event.	327
tibasSpaceMemberEvent_Free()	Frees a specified memberEvent object.	328
Member Event Operations		
tibasMemberEvent_GetType()	Returns the event type for a specified member event.	329
tibasMemberEvent_GetManagementRole()	Returns the management role for the member that generated a specified member event.	330
tibasMemberEvent_GetMember()	Returns the member ID for a specified member event.	331
tibasMemberEvent_Free()	Frees a specified memberEvent object.	332
Remote Space Member Operations		
tibasSpaceRemoteMemberEvent_GetType()	Returns the event type for a specified remote member event.	333
tibasSpaceRemoteMemberEvent_GetSpaceName()	Returns the space name for a specified remote member event.	334
tibasSpaceRemoteMemberEvent_GetRemoteMember()	Returns the member ID for a specified remote member event.	335
tibasSpaceRemoteMemberEvent_GetProxyMember()	Returns the proxy member ID for a specified remote member event.	336
tibasSpaceRemoteMemberEvent_Free()	Frees the memberEvent object for a specified RemoteMemberEvent object.	341

tibasSpaceEvent_GetTuple()

Function

Declaration

```
tibas_status tibasSpaceEvent_GetTuple(  
    tibasSpaceEvent spaceEvent,  
    tibasTuple*      tuple)
```

Purpose

Returns the tuple that generated a specified space event.

Parameters

Parameter	Description
spaceEvent	The TIBCO ActiveSpaces entity on which the function is invoked.
tuple	The tuple associated with the event.

Remarks

Use the `tibasSpaceEvent_GetTuple()` function to return the tuple associated with a `spaceEvent`. This is useful when processing events from an event browser.

See Also

[tibasSpaceEvent_GetType\(\)](#)

tibasSpaceEvent_GetType()

Function

Declaration `tibas_status` `tibasSpaceEvent_GetType`(
 `tibasSpaceEvent` `spaceEvent`,
 `tibas_eventType*` `eventType`)

Purpose Returns the type of an event generated by ActiveSpaces.

Parameters	Parameter	Description
	<code>spaceEvent</code>	Specify the <code>spaceEvent</code> for which you want to return the event type.
	<code>eventType</code>	The type of event. Possible values are <code>TIBAS_EVENT_PUT</code> , <code>TIBAS_EVENT_TAKE</code> , <code>TIBAS_EVENT_EXPIRE</code> , <code>TIBAS_EVENT_SEED</code> , or <code>TIBAS_EVENT_UNSEED</code> .

Remarks Use the `tibasSpaceEvent_GetType()` function to return the event type for a specified space event. This is useful when implementing a callback function used with a space browser.

The `eventType` parameter can return the following values:

- `TIBAS_EVENT_PUT` when a tuple is inserted, overwritten, or updated.
- `TIBAS_EVENT_TAKE` when a tuple is taken or removed.
- `TIBAS_EVENT_EXPIRE` when a tuple reaches the end of its time to live and expires from the space.
- `TIBAS_EVENT_SEED` when there is redistribution after a seeder joins or leaves, and the local node is seeding or unseeding. This is only applicable if the listener distribution scope is `SEEDED`.
- `TIBAS_EVENT_UNSEED` when there is redistribution after a seeder joins or leaves, and the local node is seeding or unseeding. This is only applicable if the listener’s distribution scope is `SEEDED`.


See Also [tibasListener_Create\(\)](#), [tibasSpaceEvent_GetTuple\(\)](#)

tibasSpaceEvent_GetSpace()

Function

Declaration `tibas_status` `tibasSpaceEvent_GetSpace`
 (`tibasSpaceEvent` `event`,
 `tibasSpace*` `space`)

Purpose Returns a Space object that identifies the space from which a space event was generated.



Note that invoking this function will increase the reference count of the returned space object. You should therefore call `tibasSpace_Free` when done processing the event.

Parameters	Parameter	Description
	<code>event</code>	Specify the <code>spaceEvent</code> for which you want to return the associated space.
	<code>space</code>	The space returned by the function.

Remarks Use the `tibasSpaceEvent_GetSpace()` function to return the name of the space that generated a specified event.

See Also [tibasSpaceEvent_GetTuple\(\)](#), [tibasSpaceEvent_GetType\(\)](#)

tibasSpaceEvent_Free()

Function

Declaration	<code>tibas_status</code> <code>tibasSpaceEvent_Free</code> (<code>tibasSpaceEvent*</code> <code>spaceEvent</code>)					
Purpose	Destroys allocated resources and frees memory.					
Parameters	<table><tr><th>Parameter</th><th>Description</th></tr><tr><td><code>spaceEvent</code></td><td>Specify the <code>spaceEvent</code> for which resources are to be freed.</td></tr></table>		Parameter	Description	<code>spaceEvent</code>	Specify the <code>spaceEvent</code> for which resources are to be freed.
Parameter	Description					
<code>spaceEvent</code>	Specify the <code>spaceEvent</code> for which resources are to be freed.					
Remarks	Use the <code>tibasSpaceEvent_Free()</code> function to free the resources and memory used by a specified <code>spaceEvent</code> .					
See Also	tibasRemoteMemberEvent_Free() , tibasSpaceMemberEvent_Free() , tibasMemberEvent_Free()					

tibasSpaceEvent_HasOldTuple()

Function

Declaration	<code>tibas_boolean</code> tibasSpaceEvent_HasOldTuple(tibasSpaceEvent spaceEvent)	
Purpose	Checks whether a specified event has an existing tuple associated with it.	
Parameters	Parameter	Description
	spaceEvent	Specifies the name of the event to check.
Remarks	<p>Use the <code>tibasSpaceEvent_HasOldTuple()</code> function to determine whether a specified <code>spaceEvent</code> has a tuple associated with it.</p> <p>Returns <code>TIBAS_TRUE</code> if the event has a tuple. Returns <code>TIBAS_FALSE</code> if the event does not have an existing tuple.</p>	
See Also	tibasSpaceEvent_GetTuple() , tibasSpaceEvent_GetType()	

tibasSpaceEvent_GetOldTuple()

Function

Declaration	<code>tibas_status</code> tibasSpaceEvent_GetOldTuple(tibasSpaceEvent spaceEvent, tibasTuple* oldTuple)	
Purpose	For a put event, returns the tuple associated with the event and updates the hash table for the space.	
Parameters	Parameter	Description
	spaceEvent	Specify the event
	oldTuple	Returns the tuple associated with a PUT event.
Remarks	<p>Use the <code>tibasSpaceEvent_GetOldTuple()</code> function to return the tuple associated with a put event.</p> <p>Calling <code>tibasSpaceEvent_GetOldTuple()</code> also causes the hash table for the space to be updated.</p>	
See Also	tibasSpaceEvent_GetTuple() , tibasSpaceEvent_GetOldTuple()	

tibasSpaceEvent_GetMetaspaceName()

Function

Declaration `tibas_status` tibasSpaceEvent_GetMetaspaceName(
 `tibasSpaceEvent` spaceEvent,
 `char**` metaspaceName)

Purpose Returns the metaspace name that is associated with a specified event.

Parameters	Parameter	Description
	spaceEvent	The space event that is to be queried.
	metaspaceName	Returns the metaspace name associated with the event.

Remarks Use the `tibasSpaceEvent_GetMetaspaceName()` function to return the metaspace associated with a specified space event.

See Also [tibasSpaceEvent_GetSpace\(\)](#)

tibasSpaceEvent_GetSpaceName()

Function

Declaration `tibas_status` tibasSpaceEvent_GetSpaceName(
 tibasSpaceEvent spaceEvent,
 char** spaceName)

Purpose Returns the space name associated with a specified space event.

Parameters	Parameter	Description
	spaceEvent	Specify a space event for which you want to determine a space name.
	spaceName	Returns the space name associated with the specified space event.

Remarks Use the tibasSpaceEvent_GetSpaceName() function to return the space name for the space associated with a specified space event.

See Also [tibasSpaceEvent_GetMetaspaceName\(\)](#)

tibasSpaceMemberEvent_GetType()

Function

Declaration `tibas_status tibasSpaceMemberEvent_GetType(
 tibasSpaceMemberEvent spaceMemberEvent,
 tibas_memberEventType* memberEventType);`

Purpose Returns the event type for a specified space member event.

Parameters	Parameter	Description
	spaceMemberEvent	Specify the spaceMemberEvent structure that identifies the event for which to return an event type.
	memberEventType	Returns the space member event type.

Remarks Use the `tibasSpaceMemberEvent_GetType()` function to return the event type for a specified space member event.

The space member event type can be one of the following:

- **TIBAS_MEMBER_EVENT_JOIN** Indicates that the member has joined the space.
- **TIBAS_MEMBER_EVENT_UPDATE** Indicates that the member is being updated.
- **TIBAS_MEMBER_EVENT_LEAVE** Indicates that the member has left the space.

See Also [tibasMemberEvent_GetType\(\)](#), [tibasSpaceMemberEvent_GetType\(\)](#),
[tibasSpaceRemoteMemberEvent_GetType\(\)](#),
[tibasRemoteMemberEvent_GetType\(\)](#)

tibasSpaceMemberEvent_GetSpaceName()

Function

Declaration `tibas_status` tibasSpaceMemberEvent_GetSpaceName(
 tibasSpaceMemberEvent spaceMemberEvent,
 char** spaceName)

Purpose Returns the space name for a specified space member event.

Parameters	Parameter	Description
	spaceMemberEvent	Specify the SpaceMemberEvent object for which you want to return the member name.
	spaceName	Returns the space name for the member event.

Remarks Use the `tibasSpaceMemberEvent_GetSpaceName()` function to return the space name for a specified space member event.

See Also [tibasSpaceMemberEvent_GetType\(\)](#),
 [tibasSpaceMemberEvent_GetDistributionRole\(\)](#),
 [tibasSpaceMemberEvent_GetMember\(\)](#)

tibasSpaceMemberEvent_GetDistributionRole()

Function

Declaration `tibas_status tibasSpaceMemberEvent_GetDistributionRole(
 tibasSpaceMemberEvent spaceMemberEvent,
 tibas_distributionRole* distributionRole)`

Purpose Returns the distribution role for a specified space member event.

Parameters	Parameter	Description
	spaceMemberEvent	Specify the SpaceMemberEvent object for which you want to return the member name.
	distributionRole	Returns the distribution role for the specified space member event.

Remarks Use the `tibasSpaceMemberEvent_GetDistributionRole()` function to return the distribution role for a specified space member event.

The distribution role can be one of the following:

- **TIBAS_DISTRIBUTION_ROLE_SEEDER** Indicates that the space member is a seeder—an application that participates in the storing of data in the space, and can read and write data. When seeder applications join or leave the space, ActiveSpaces redistributes the data in the space as necessary to maintain even data distribution.
- **TIBAS_DISTRIBUTION_ROLE_LEECH** Indicates that the space member is a leech—an application that passively participates in the space and does not read and write data or cause redistribution of space data when it joins or leaves the space.
- **TIBAS_DISTRIBUTION_ROLE_NONE** Indicates that the specified member has not joined the space.

See Also [tibasSpaceMemberEvent_GetType\(\)](#),
[tibasSpaceMemberEvent_GetSpaceName\(\)](#)[tibasSpaceMemberEvent_GetDistributionRole\(\)](#), [tibasSpaceMemberEvent_GetMember\(\)](#),

tibasSpaceMemberEvent_GetMember()

Function

Declaration

```
tibas_status tibasSpaceMemberEvent_GetMember(  
    tibasSpaceMemberEvent spaceMemberEvent,  
    tibasMember*          member)
```

Purpose

Returns the member ID for a specified space member event.

Parameters

Parameter	Description
spaceMemberEvent	Specify the SpaceMemberEvent object for which you want to return the member.
member	Returns the member ID for the space member event.

Remarks

Use the `tibasSpaceMemberEvent_GetMember()` function to return the member ID for a specified space member event.

See Also

[tibasSpaceMemberEvent_GetType\(\)](#),
[tibasSpaceRemoteMemberEvent_GetSpaceName\(\)](#),
[tibasMemberEvent_GetManagementRole\(\)](#),
[tibasSpaceRemoteMemberEvent_GetRemoteMember\(\)](#)

tibasSpaceMemberEvent_Free()

Function

Declaration	<code>tibas_status tibasSpaceMemberEvent_Free(tibasSpaceMemberEvent* spaceMemberEvent)</code>	
Purpose	Frees a specified SpaceMemberEvent object.	
Parameters	Parameter	Description
	spaceMemberEvent	Specify the SpaceMemberEvent object to free.
Remarks	Use the <code>tibasSpaceMemberEvent_Free()</code> function to free a specified <code>spaceMemberEvent</code> object.	
See Also	tibasSpaceMemberEvent_GetType() , tibasSpaceRemoteMemberEvent_GetSpaceName() , tibasMemberEvent_GetManagementRole() , tibasSpaceMemberEvent_GetDistributionRole() , tibasSpaceEvent_Free() , tibasSpaceMemberEvent_Free() , tibasMemberEvent_Free()	

tibasMemberEvent_GetType()

Function

Declaration

```
tibas_status tibasMemberEvent_GetType(  
    tibasMemberEvent      memberEvent,  
    tibas_memberEventType* memberEventType)
```

Purpose

Returns the event type for a specified member event.

Parameters

Parameter	Description
memberEvent	Specify the memberEvent object for which you want to return the event type.
memberEventType	Returns the event type.

Remarks

Use the `tibasMemberEvent_GetType()` function to return the event type for a specified member event.

The member event type can be one of the following:

- **TIBAS_MEMBER_EVENT_JOIN** Indicates that the member has joined the metaspace.
- **TIBAS_MEMBER_EVENT_UPDATE** Indicates that the member is being updated.
- **TIBAS_MEMBER_EVENT_LEAVE** Indicates that the member has left the metaspace.

See Also

[tibasMemberEvent_GetType\(\)](#), [tibasSpaceMemberEvent_GetType\(\)](#), [tibasSpaceRemoteMemberEvent_GetType\(\)](#), [tibasRemoteMemberEvent_GetType\(\)](#), [tibasMemberEvent_GetManagementRole\(\)](#), [tibasMemberEvent_GetMember\(\)](#),

tibasMemberEvent_GetManagementRole()

Function

Declaration

`tibas_status` `tibasMemberEvent_GetManagementRole`(
 `tibasMemberEvent` `memberEvent`,
 `tibas_managementRole*` `managementRole`

Purpose

Returns the management role for the member that generated a specified member event.

Parameters

Parameter	Description
<code>memberEvent</code>	Specify the <code>memberEvent</code> object for which you want to return the member management role.
<code>managementRole</code>	Returns the management role for the member that generated the event.

Remarks

Use the `tibasMemberEvent_GetManagementRole()` function to return the management role for the member that generated a specified member event.

The management role can be one of the following:

- **TIBAS_MANAGEMENT_ROLE_NONE** The member has no management role.
- **TIBAS_MANAGEMENT_ROLE_MEMBER** The member is simply a member of the Metaspace.
- **TIBAS_MANAGEMENT_ROLE_MANAGER** The member is currently managing membership for the metaspace.

See Also

[tibasMemberEvent_GetType\(\)](#), [tibasMemberEvent_GetMember\(\)](#)

tibasMemberEvent_GetMember()

Function

Declaration `tibas_status` tibasMemberEvent_GetMember(
 tibasMemberEvent memberEvent,
 tibasMember* member)

Purpose Returns the member ID for a specified member event.

Parameters	Parameter	Description
	memberEvent	Specify the memberEvent object for which you want to return the member ID.
	member	Returns the member ID for the member that generated the event.

Remarks Use the `tibasMemberEvent_GetMember()` function to return the member ID for a the member that generated a specified member event.

See Also [tibasMemberEvent_GetType\(\)](#), [tibasMemberEvent_GetManagementRole\(\)](#)

tibasMemberEvent_Free()

Function

Declaration `tibas_status tibasMemberEvent_Free(
 tibasMemberEvent* memberEvent);;`

Purpose Frees a specified memberEvent object.

Parameters	Parameter	Description
	memberEvent	Specify the memberEvent object that you want to free.

Remarks Use the tibasMemberEvent_Free() function to free a specified memberEventObject.

See Also [tibasMemberEvent_GetType\(\)](#), [tibasMemberEvent_GetManagementRole\(\)](#), [tibasMemberEvent_GetMember\(\)](#), [tibasSpaceEvent_Free\(\)](#), [tibasSpaceMemberEvent_Free\(\)](#), [tibasMemberEvent_Free\(\)](#)

tibasSpaceRemoteMemberEvent_GetType()

Function

Declaration

```
tibas_status tibasSpaceRemoteMemberEvent_GetType(  
    tibasSpaceRemoteMemberEvent memberEvent,  
    tibas_memberEventType*      memberEventType);
```

Purpose

Returns the event type for a specified space remote member event.

Parameters

Parameter	Description
memberEvent	Specify the memberEvent object for which you want to return the event type.
memberEventType	Returns the event type.

Remarks

Use the `tibasSpaceRemoteMemberEvent_GetType()` function to return the event type for a specified space remote member event.

The remote space member event type can be one of the following:

- **TIBAS_MEMBER_EVENT_JOIN** Indicates that the remote space member has joined the space.
- **TIBAS_MEMBER_EVENT_LEAVE** Indicates that the remote space member has left the space.

See Also

[tibasMemberEvent_GetType\(\)](#), [tibasSpaceMemberEvent_GetType\(\)](#), [tibasRemoteMemberEvent_GetType\(\)](#)

tibasSpaceRemoteMemberEvent_GetSpaceName()

Function

Declaration

```
tibas_status tibasSpaceRemoteMemberEvent_GetSpaceName(  
    tibasSpaceRemoteMemberEvent spaceMemberEvent,  
    char** spaceName)
```

Purpose

Returns the space name for a specified remote member event.

Parameters

Parameter	Description
spaceMemberEvent	Specify the SpaceMemberEvent object for which you want to return the member name.
spaceName	Returns the space name for the member event.

Remarks

Use the `tibasSpaceRemoteMemberEvent_GetSpaceName()` function to return the space name for a specified remote space member event.

See Also

[tibasSpaceEvent_GetMetaspaceName\(\)](#), [tibasSpaceEvent_GetSpaceName\(\)](#),

tibasSpaceRemoteMemberEvent_GetRemoteMember()

Function

Declaration `tibas_status` tibasSpaceRemoteMemberEvent_GetRemoteMember(
 tibasSpaceRemoteMemberEvent spaceMemberEvent,
 tibasMember* member);

Purpose Returns the member ID for a specified remote member event.

Parameters	Parameter	Description
	spaceMemberEvent	Specify the SpaceMemberEvent object for which you want to return the member.
	member	Returns the member ID for the remote member event.

Remarks Use the `tibasSpaceRemoteMemberEvent_GetRemoteMember()` function to return the member ID for a specified remote space member event.

See Also [tibasSpaceMemberEvent_GetMember\(\)](#), [tibasMemberEvent_GetMember\(\)](#), [tibasRemoteMemberEvent_GetRemoteMember\(\)](#)

tibasSpaceRemoteMemberEvent_GetProxyMember()

Function

- Declaration**

```
tibas_status tibasSpaceRemoteMemberEvent_GetProxyMember(  
    tibasSpaceRemoteMemberEvent spaceMemberEvent,  
    tibasMember* member)
```
- Purpose** Returns the proxy member ID for a specified remote space member event.

Parameters

Parameter	Description
spaceMemberEvent	Specify the SpaceMemberEvent object for which you want to return the proxy member.
member	Returns the proxy member ID for the remote space member event.

- Remarks** Use the `tibasRemoteMemberEvent_GetProxyMember()` function to return the proxy member ID for a specified remote space member event.
- See Also** [tibasRemoteMemberEvent_GetProxyMember\(\)](#)

tibasSpaceRemoteMemberEvent_Free()

Function

Declaration	<code>tibas_status</code> tibasSpaceRemoteMemberEvent_Free(tibasSpaceRemoteMemberEvent* memberEvent)	
Purpose	Frees a specified memberEvent object for a remote member event.	
Parameters	Parameter	Description
	memberEvent	Specify the memberEvent object that you want to free.
Remarks	Use the <code>tibasSpaceRemoteMemberEvent_Free()</code> function to free a memberEvent object for a specified remote member event.	

tibasRemoteMemberEvent_GetType()

Function

Declaration `tibas_status tibasRemoteMemberEvent_GetType(
 tibasRemoteMemberEvent memberEvent,
 tibas_memberEventType* memberEventType);`

Purpose Returns the event type for a specified remote member event.

Parameters	Parameter	Description
	memberEvent	Specify the memberEvent object for which you want to return the event type.
	memberEventType	Returns the event type.

Remarks Use the tibasRemoteMemberEvent_GetType() function to return the event type for a specified remote member event.

The remote member event type can be one of the following:

- **TIBAS_MEMBER_EVENT_JOIN** Indicates that the remote space member has joined the metaspace.
- **TIBAS_MEMBER_EVENT_LEAVE** Indicates that the remote space member has left the metaspace.

See Also [tibasMemberEvent_GetType\(\)](#), [tibasSpaceMemberEvent_GetType\(\)](#), [tibasRemoteMemberEvent_GetType\(\)](#)

tibasRemoteMemberEvent_GetRemoteMember()

Function

Declaration

```
tibas_status tibasRemoteMemberEvent_GetRemoteMember(  
    tibasRemoteMemberEvent memberEvent,  
    tibasMember*            member)
```

Purpose

Returns the member ID for a specified remote member event.

Parameters

Parameter	Description
spaceMemberEvent	Specify the SpaceMemberEvent object for which you want to return the member.
member	Returns the member ID for the remote member event.

Remarks

Use the `tibasRemoteMemberEvent_GetRemoteMember()` function to return the member ID for a specified remote member event.

See Also

[tibasMemberEvent_GetMember\(\)](#), [tibasSpaceMemberEvent_GetMember\(\)](#), [tibasSpaceRemoteMemberEvent_GetRemoteMember\(\)](#)

tibasRemoteMemberEvent_GetProxyMember()

Function

Declaration `tibas_status` tibasRemoteMemberEvent_GetProxyMember(
 tibasRemoteMemberEvent memberEvent,
 tibasMember* member)

Purpose Returns the proxy member ID for a specified remote member event.

Parameters	Parameter	Description
	memberEvent	Specify the memberEvent object for which you want to return the proxy member.
	member	Returns the proxy member ID for the remote member event.

Remarks Use the `tibasRemoteMemberEvent_GetProxyMember()` function to return the proxy member ID for a specified remote member event.

See Also [tibasRemoteMemberEvent_GetProxyMember\(\)](#)

tibasRemoteMemberEvent_Free()

Function

Declaration `tibas_status` `tibasRemoteMemberEvent_Free(`
 `tibasRemoteMemberEvent* memberEvent)`

Purpose Frees the memberEvent object for a specified RemoteMemberEvent object.

Parameters	Parameter	Description
	memberEvent	Specify the memberEvent object that you want to free.

Remarks Use the `tibasRemoteMemberEvent_Free()` object to free a specified memberEvent object for a remote member event.

See Also [tibasSpaceEvent_Free\(\)](#), [tibasSpaceMemberEvent_Free\(\)](#),
[tibasMemberEvent_Free\(\)](#)

Chapter 18 **Tuple**

This chapter explains the tuple operations.

A tuple is similar to a row in a database table: it is a container for data. More specifically, it is a sequence of named elements called fields (similar to the columns in a database table) which contain values of a specific type. Each tuple in a space represents a set of related data.

Topics

- [Tuple Operations](#)

Tuple Operations

The following table lists the Tuple operations:

Table 23 Tuple

Function or Type	Description	Page
tibasTuple_Clone()	Returns a copy of a specified tuple instance.	348
tibasTuple_Deserialize()	Creates a tuple object from a serialized form of a tuple.	375
tibasTuple_Eval()	Evaluates if a specified tuple passes a specified filter.	381
tibasTuple_Exists()	Returns a boolean value indicating whether the specified field exists.	377
tibasTuple_Free()	Destroys allocated resources and frees memory.	382
tibasTuple_GetBlob()	Returns the binary large object (BLOB) value stored in the field with the specified name.	358
tibasTuple_GetBoolean()	Returns the boolean value stored in the field with the specified name.	349
tibasTuple_GetChar()	Returns the char value stored in a field with the specified name.	350
tibasTuple_GetDateTime()	Returns the datetime value stored in the field with the specified name.	357
tibasTuple_GetDouble()	Returns the double value stored in the field with the specified name.	355
tibasTuple_GetFieldNames()	Returns an array of names of fields inside the specified tuple.	378
tibasTuple_GetFieldType()	Returns the type of the specified field.	379
tibasTuple_GetFloat()	Returns the float value stored in the field with the specified name.	354
tibasTuple_GetInt()	Returns the int value stored in the field with the specified name.	352

Table 23 Tuple

Function or Type	Description	Page
tibasTuple_GetLong()	Returns the long value stored in the field with the specified name.	353
tibasTuple_GetShort()	Returns the short value stored in the field with the specified name.	351
tibasTuple_GetString()	Returns the string value stored in the field with the specified name.	356
tibasTuple_IsNull()	Returns a boolean value indicating whether or not the specified tuple field is <code>null</code> (such as when it is not present in the tuple).	376
tibasTuple_PutAll()	Puts a collection of fields into the tuple.	380
tibasTuple_PutBlob()	Stores a binary large object (BLOB) value in a specified field in a specified tuple.	369
tibasTuple_PutBoolean()	Stores a boolean value in a specified field with the specified name.	359
tibasTuple_PutChar()	Stores a char value in the field with the specified name.	360
tibasTuple_PutDateTime()	Stores a datetime value in a specified field in a specified tuple.	367
tibasTuple_PutDouble()	Stores a double value in a specified field in a specified tuple.	365
tibasTuple_PutFloat()	Stores a float value in the field with the specified name.	364
tibasTuple_PutInt()	Stores an integer value in the field with the specified name.	362
tibasTuple_PutLong()	Stores a long value in a specified tuple with a specified fieldname.	363
tibasTuple_PutShort()	Stores a short value in the field with the specified name.	361

Table 23 Tuple

Function or Type	Description	Page
tibasTuple_PutString()	Stores a string value in a specified field in a specified tuple.	366
tibasTuple_Serialize()	Serializes the tuple into a string of bytes.	374
tibasTuple_Size()	Returns the number of fields contained in a specified tuple.	371
tibasTuple_Clear()	Clears the data for the specified tuple.	372
tibasTuple_Remove()	Removes a specified field from a specified tuple.	370
tibasTuple_ToString()	Creates a string representation of the specified tuple.	373

tibasTuple_Create()

Function

Declaration `tibas_status` `tibasTuple_Create`
 (`tibasTuple*` `tuple`)

Purpose Returns a new, empty tuple instance.

Parameters	Parameter	Description
	<code>tuple</code>	The new empty tuple returned by the function.

Remarks Use the `tibasTuple_Create()` function to create a new tuple instance.

See Also [tibasTuple_Remove\(\)](#), [tibasTuple_Size\(\)](#), [tibasTuple_Clear\(\)](#)

tibasTuple_Clone()

Function

Declaration `tibas_status` `tibasTuple_Clone`
 (`tibasTuple*` `tuple`
 `tibasTuple*` `clone`)

Purpose Returns a copy of a specified tuple instance.

Parameters	Parameter	Description
	<code>tuple</code>	The new empty tuple returned by the function.

Remarks Use the `tibasTuple_Clone()` function to create a copy of an existing tuple instance.

See Also [tibasTuple_Remove\(\)](#), [tibasTuple_Size\(\)](#), [tibasTuple_Clear\(\)](#)

tibasTuple_GetBoolean()

Function

Declaration

```
tibas_status tibasTuple_GetBoolean
(tibasTuple tuple,
tibas_boolean* value,
const char* fieldname)
```

Purpose

Returns the boolean value stored in the field with the specified name.

Parameters

Parameter	Description
tuple	Specify the tuple for which you want to return a boolean value.
value	The boolean value returned by the function.
fieldname	The fieldname.

Remarks

Use the `tibasTuple_GetBoolean()` function to return the value of a boolean field in a specified tuple.

The function returns `TIBAS_TRUE` if the boolean value is 1, and returns `TIBAS_FALSE` if the boolean value is 0.

See Also

[tibasTuple_GetChar\(\)](#), [tibasTuple_GetShort\(\)](#), [tibasTuple_GetInt\(\)](#), [tibasTuple_GetLong\(\)](#), [tibasTuple_GetLong\(\)](#), [tibasTuple_GetFloat\(\)](#), [tibasTuple_GetFloat\(\)](#), [tibasTuple_GetDouble\(\)](#), [tibasTuple_GetString\(\)](#), [tibasTuple_GetDateTime\(\)](#), [tibasTuple_GetBlob\(\)](#)

tibasTuple_GetChar()

Function

Declaration `tibas_status` `tibasTuple_GetChar`
 (`tibasTuple` `tuple`,
 `char*` `value`,
 `const char*` `fieldname`)

Purpose Returns the char value stored in a field with the specified name.

Parameter	Description
<code>tuple</code>	Specify the tuple for which you want to return a character value.
<code>value</code>	The char value returned by the function.
<code>fieldname</code>	The field name.

Remarks Use the `tibasTuple_GetChar()` function to return the char value stored in a specified field in a specified tuple.

See Also [tibasTuple_GetBoolean\(\)](#), [tibasTuple_GetShort\(\)](#), [tibasTuple_GetInt\(\)](#),
[tibasTuple_GetLong\(\)](#), [tibasTuple_GetLong\(\)](#), [tibasTuple_GetFloat\(\)](#),
[tibasTuple_GetFloat\(\)](#), [tibasTuple_GetDouble\(\)](#), [tibasTuple_GetString\(\)](#),
[tibasTuple_GetDateTime\(\)](#), [tibasTuple_GetBlob\(\)](#)

tibasTuple_GetShort()

Function

Declaration

```
tibas_status tibasTuple_GetShort
(tibasTuple tuple,
tibas_short* value,
const char* fieldname)
```

Purpose

Returns the short value stored in the field with the specified name.

Parameters

Parameter	Description
tuple	Specify the tuple for which you want to return a short value.
value	The short value returned by the function.
fieldname	The fieldname.

Remarks

Use the `tibasTuple_GetShort()` function to return the short value stored in a specified field in a specified tuple.

See Also

[tibasTuple_GetChar\(\)](#), [tibasTuple_GetBoolean\(\)](#), [tibasTuple_GetInt\(\)](#), [tibasTuple_GetLong\(\)](#), [tibasTuple_GetLong\(\)](#), [tibasTuple_GetFloat\(\)](#), [tibasTuple_GetFloat\(\)](#), [tibasTuple_GetDouble\(\)](#), [tibasTuple_GetString\(\)](#), [tibasTuple_GetDateTime\(\)](#), [tibasTuple_GetBlob\(\)](#)

tibasTuple_GetInt()

Function

Declaration `tibas_status` `tibasTuple_GetInt`
 (`tibasTuple` `tuple`,
 `tibas_int*` `value`,
 `const char*` `fieldname`)

Purpose Returns the int value stored in the field with the specified name.

Parameters	Parameter	Description
	<code>tuple</code>	Specify the tuple for which you want to return an integer value.
	<code>value</code>	The int value returned by the function.
	<code>fieldname</code>	The fieldname.

Remarks Use the `tibasTuple_GetInt()` function to return the integer value stored in a specified field in a specified tuple.

See Also [tibasTuple_GetChar\(\)](#), [tibasTuple_GetShort\(\)](#), [tibasTuple_GetBoolean\(\)](#),
[tibasTuple_GetLong\(\)](#), [tibasTuple_GetLong\(\)](#), [tibasTuple_GetFloat\(\)](#),
[tibasTuple_GetFloat\(\)](#), [tibasTuple_GetDouble\(\)](#), [tibasTuple_GetString\(\)](#),
[tibasTuple_GetDateTime\(\)](#), [tibasTuple_GetBlob\(\)](#)

tibasTuple_GetLong()

Function

Declaration `tibas_status` `tibasTuple_GetLong`
 (`tibasTuple` `tuple`,
 `tibas_long*` `value`,
 `const char*` `fieldname`)

Purpose Returns the long value stored in the field with the specified name.

Parameters	Parameter	Description
	<code>tuple</code>	Specify the tuple for which you want to return a long value.
	<code>value</code>	The long value returned by the function.
	<code>fieldname</code>	The fieldname.

Remarks Use the `tibasTuple_GetLong()` function to return the long value stored in a specified field in a specified tuple.

See Also [tibasTuple_GetChar\(\)](#), [tibasTuple_GetShort\(\)](#), [tibasTuple_GetInt\(\)](#),
[tibasTuple_GetBoolean\(\)](#), [tibasTuple_GetLong\(\)](#), [tibasTuple_GetFloat\(\)](#),
[tibasTuple_GetFloat\(\)](#), [tibasTuple_GetDouble\(\)](#), [tibasTuple_GetString\(\)](#),
[tibasTuple_GetDateTime\(\)](#), [tibasTuple_GetBlob\(\)](#)

tibasTuple_GetFloat()

Function

Declaration `tibas_status tibasTuple_GetFloat
(tibasTuple tuple,
tibas_float* value,
const char* fieldname)`

Purpose Returns the float value stored in the field with the specified name.

Parameters	Parameter	Description
	tuple	Specify the tuple for which you want to return a boolean value.
	value	The float value returned by the function.
	fieldname	The fieldname.

Remarks Use the `tibasTuple_GetFloat()` function to return the float value stored in a specified field in a specified tuple.

See Also [tibasTuple_GetChar\(\)](#), [tibasTuple_GetShort\(\)](#), [tibasTuple_GetInt\(\)](#),
[tibasTuple_GetLong\(\)](#), [tibasTuple_GetLong\(\)](#), [tibasTuple_GetBoolean\(\)](#),
[tibasTuple_GetBoolean\(\)](#), [tibasTuple_GetDouble\(\)](#), [tibasTuple_GetString\(\)](#),
[tibasTuple_GetDateTime\(\)](#), [tibasTuple_GetBlob\(\)](#)

tibasTuple_GetDouble()

Function

Declaration `tibas_status` tibasTuple_GetDouble
 (`tibasTuple` `tuple`,
 `tibas_double*` `value`,
 `const char*` `fieldname`)

Purpose Returns the double value stored in the field with the specified name.

Parameters

Parameter	Description
<code>tuple</code>	Specify the tuple for which you want to return a double value.
<code>value</code>	The double value returned by the function.
<code>fieldname</code>	The fieldname.

Remarks Use the `tibasTuple_GetDouble()` function to return the double value stored in a specified field in a specified tuple.

See Also [tibasTuple_GetChar\(\)](#), [tibasTuple_GetShort\(\)](#), [tibasTuple_GetInt\(\)](#),
[tibasTuple_GetLong\(\)](#), [tibasTuple_GetLong\(\)](#), [tibasTuple_GetFloat\(\)](#),
[tibasTuple_GetFloat\(\)](#), [tibasTuple_GetBoolean\(\)](#), [tibasTuple_GetString\(\)](#),
[tibasTuple_GetDateTime\(\)](#), [tibasTuple_GetBlob\(\)](#)

tibasTuple_GetString()

Function

Declaration `tibas_status` `tibasTuple_GetString`
 (`tibasTuple` `tuple`,
 `char**` `value`,
 `const char*` `fieldname`)

Purpose Returns the string value stored in the field with the specified name.

Parameter	Description
<code>tuple</code>	Specify the tuple for which you want to return a string value.
<code>value</code>	The string value returned by the function. The program must not modify or free the string, as it is owned by the tuple object.
<code>fieldname</code>	The field name.

Remarks Use the `tibasTuple_GetString()` function to return the string value stored in a specified field in a specified tuple.

See Also [tibasTuple_GetChar\(\)](#), [tibasTuple_GetShort\(\)](#), [tibasTuple_GetInt\(\)](#),
[tibasTuple_GetLong\(\)](#), [tibasTuple_GetLong\(\)](#), [tibasTuple_GetFloat\(\)](#),
[tibasTuple_GetFloat\(\)](#), [tibasTuple_GetDouble\(\)](#), [tibasTuple_GetBoolean\(\)](#),
[tibasTuple_GetDateTime\(\)](#), [tibasTuple_GetBlob\(\)](#)

tibasTuple_GetDateTime()

Function

Declaration `tibas_status` `tibasTuple_GetDateTime`
 (`tibasTuple` `tuple`,
 `tibas_DateTime*` `dateTime`,
 `const char*` `fieldname`)

Purpose Returns the datetime value stored in the field with the specified name.

Parameters	Parameter	Description
	<code>tuple</code>	Specify the tuple for which you want to return a date and time value.
	<code>dateTime</code>	Returns a pointer to a <code>_tibas_DateTime</code> struct that holds the datetime information returned by the function.
	<code>fieldname</code>	The fieldname.

Remarks Use the `tibasTuple_GetDateTime()` function to return the datetime value stored in a specified field in a specified tuple.

The function returns a pointer to a `_tibas_dateTime` structure, The `_tibasDateTime` struct is defined as follows:

```
struct _tibasDateTime {
    tibas_long sec;
    tibas_int nsec;
};
```

The members in the `_tibas_dateTime` struct are defined as follows:

- **sec** Indicates the time, in seconds, since the UNIX EPOCH time (1970-01-01 00:00:00 +0000).
- **nsec** Indicates the time, in nanoseconds, since the UNIX EPOCH time (1970-01-01 00:00:00 +0000). This field provides is extra timing information in addition to the number of seconds since the EPOCH time; usually this is set to 0.

See Also [tibasTuple_GetChar\(\)](#), [tibasTuple_GetShort\(\)](#), [tibasTuple_GetInt\(\)](#), [tibasTuple_GetLong\(\)](#), [tibasTuple_GetLong\(\)](#), [tibasTuple_GetFloat\(\)](#), [tibasTuple_GetFloat\(\)](#), [tibasTuple_GetDouble\(\)](#), [tibasTuple_GetString\(\)](#), [tibasTuple_GetBoolean\(\)](#), [tibasTuple_GetBlob\(\)](#), [tibasTuple_PutDateTime\(\)](#)

tibasTuple_GetBlob()

Function

Declaration `tibas_status` `tibasTuple_GetBlob`
 (`tibasTuple` `tuple`,
 `char**` `value`,
 `tibas_int*` `length`,
 `const char*` `fieldname`)

Purpose Returns the binary large object (BLOB) value stored in the field with the specified name.

Parameter	Parameter	Description
	<code>tuple</code>	Specify the tuple for which you want to return a blob value.
	<code>value</code>	The blob value returned by the function. The program must not modify or free the blob.
	<code>length</code>	The length of the return value.
	<code>fieldname</code>	The fieldname.

Remarks Use the `tibasTuple_GetBlob()` function to return the blob value stored in a specified field in a specified tuple.

See Also [tibasTuple_GetChar\(\)](#), [tibasTuple_GetShort\(\)](#), [tibasTuple_GetInt\(\)](#),
[tibasTuple_GetLong\(\)](#), [tibasTuple_GetLong\(\)](#), [tibasTuple_GetFloat\(\)](#),
[tibasTuple_GetFloat\(\)](#), [tibasTuple_GetDouble\(\)](#), [tibasTuple_GetString\(\)](#),
[tibasTuple_GetDateTime\(\)](#), [tibasTuple_GetBoolean\(\)](#)

tibasTuple_PutBoolean()

Function

Declaration `tibas_status` `tibasTuple_PutBoolean`
 `(tibasTuple tuple,`
 `const char* fieldname,`
 `tibas_boolean value)`

Purpose Stores a boolean value in a specified field with the specified name.

Parameter	Parameter	Description
	<code>tuple</code>	Specify the tuple into which you want to put a boolean value.
	<code>fieldname</code>	The fieldname.
	<code>value</code>	The boolean value to store in the tuple.

Remarks Use the `tibasTuple_PutBoolean()` function to put a specified boolean value in a specified field in a specified tuple.

See Also [tibasTuple_PutChar\(\)](#), [tibasTuple_PutShort\(\)](#), [tibasTuple_PutInt\(\)](#),
[tibasTuple_PutLong\(\)](#), [tibasTuple_PutFloat\(\)](#), [tibasTuple_PutDouble\(\)](#),
[tibasTuple_PutString\(\)](#), [tibasTuple_PutDateTime\(\)](#)[tibasTuple_PutBlob\(\)](#)

tibasTuple_PutChar()

Function

Declaration `tibas_status` `tibasTuple_PutChar`
 (`tibasTuple` `tuple`,
 `const char*` `fieldname`,
 `char` `value`)

Purpose Stores a char value in the field with the specified name.

Parameter	Parameter	Description
	<code>tuple</code>	Specify the tuple into which you want to put a char value.
	<code>fieldname</code>	The fieldname.
	<code>value</code>	The char value to store in the fieldname.

Remarks Use the `tibasTuple_PutChar()` function to put a specified boolean value in a specified field in a specified tuple.

See Also [tibasTuple_PutBoolean\(\)](#), [tibasTuple_PutShort\(\)](#), [tibasTuple_PutInt\(\)](#),
[tibasTuple_PutLong\(\)](#), [tibasTuple_PutFloat\(\)](#), [tibasTuple_PutDouble\(\)](#),
[tibasTuple_PutString\(\)](#), [tibasTuple_PutDateTime\(\)](#)[tibasTuple_PutBlob\(\)](#)

tibasTuple_PutShort()

Function

Declaration `tibas_status` `tibasTuple_PutShort`
 (`tibasTuple` `tuple`,
 `const char*` `fieldname`,
 `tibas_short` `value`)

Purpose Stores a short value in the field with the specified name.

Parameter	Parameter	Description
	<code>tuple</code>	<code>v</code>
	<code>fieldname</code>	The fieldname.
	<code>value</code>	The short value to store in the fieldname.

Remarks Use the `tibasTuple_PutShort()` function to put a specified short value in a specified field in a specified tuple.

See Also [tibasTuple_PutChar\(\)](#), [tibasTuple_PutBoolean\(\)](#), [tibasTuple_PutInt\(\)](#),
[tibasTuple_PutLong\(\)](#), [tibasTuple_PutFloat\(\)](#), [tibasTuple_PutDouble\(\)](#),
[tibasTuple_PutString\(\)](#), [tibasTuple_PutDateTime\(\)](#)[tibasTuple_PutBlob\(\)](#)

tibasTuple_PutInt()

Function

Declaration `tibas_status` `tibasTuple_PutInt`
 (`tibasTuple` `tuple`,
 `const char*` `fieldname`,
 `tibas_int` `value`)

Purpose Stores an integer value in the field with the specified name.

Parameter	Parameter	Description
	<code>tuple</code>	Specify the tuple into which you want to put an integer value.
	<code>fieldname</code>	The fieldname.
	<code>value</code>	The integer value to store in the fieldname.

Remarks Use the `tibasTuple_PutInt()` function to put a specified integer value in a specified field in a specified tuple.

See Also [tibasTuple_PutChar\(\)](#), [tibasTuple_PutShort\(\)](#), [tibasTuple_PutBoolean\(\)](#),
[tibasTuple_PutLong\(\)](#), [tibasTuple_PutFloat\(\)](#), [tibasTuple_PutDouble\(\)](#),
[tibasTuple_PutString\(\)](#), [tibasTuple_PutDateTime\(\)](#)[tibasTuple_PutBlob\(\)](#)

tibasTuple_PutLong()

Function

Declaration `tibas_status` `tibasTuple_PutLong`
 (`tibasTuple` `tuple`,
 `const char*` `fieldname`,
 `tibas_long` `value`)

Purpose Stores a long value in a specified tuple with a specified fieldname.

Parameter	Parameter	Description
	<code>tuple</code>	Specify the tuple into which you want to put a long value.
	<code>fieldname</code>	The fieldname.
	<code>value</code>	The long value to store in the fieldname.

Remarks Use the `tibasTuple_PutLong()` function to put a specified long value in a specified field in a specified tuple.

See Also [tibasTuple_PutChar\(\)](#), [tibasTuple_PutShort\(\)](#), [tibasTuple_PutInt\(\)](#),
[tibasTuple_PutBoolean\(\)](#), [tibasTuple_PutFloat\(\)](#), [tibasTuple_PutDouble\(\)](#),
[tibasTuple_PutString\(\)](#), [tibasTuple_PutDateTime\(\)](#)[tibasTuple_PutBlob\(\)](#)

tibasTuple_PutFloat()

Function

Declaration `tibas_status` `tibasTuple_PutFloat`
 (`tibasTuple` `tuple`,
 `const char*` `fieldname`,
 `tibas_float` `value`)

Purpose Stores a float value in the field with the specified name.

Parameter	Parameter	Description
	<code>tuple</code>	Specify the tuple into which you want to put a float value.
	<code>fieldname</code>	The fieldname.
	<code>value</code>	The float value to store in the fieldname.

Remarks Use the `tibasTuple_PutFloat()` function to put a specified float value in a specified field in a specified tuple.

See Also [tibasTuple_PutChar\(\)](#), [tibasTuple_PutShort\(\)](#), [tibasTuple_PutInt\(\)](#),
[tibasTuple_PutLong\(\)](#), [tibasTuple_PutBoolean\(\)](#), [tibasTuple_PutDouble\(\)](#),
[tibasTuple_PutString\(\)](#), [tibasTuple_PutDateTime\(\)](#)[tibasTuple_PutBlob\(\)](#)

tibasTuple_PutDouble()

Function

Declaration `tibas_status` `tibasTuple_PutDouble`
 (`tibasTuple` `tuple`,
 `const char*` `fieldname`,
 `tibas_double` `value`)

Purpose Stores a double value in a specified field in a specified tuple.

Parameter	Parameter	Description
	<code>tuple</code>	Specify the tuple into which you want to put a double value.
	<code>fieldname</code>	The fieldname.
	<code>value</code>	The double value to store in the fieldname.

Remarks Use the `tibasTuple_PutDouble()` function to put a specified double value in a specified field in a specified tuple.

See Also [tibasTuple_PutChar\(\)](#), [tibasTuple_PutShort\(\)](#), [tibasTuple_PutInt\(\)](#),
[tibasTuple_PutLong\(\)](#), [tibasTuple_PutFloat\(\)](#), [tibasTuple_PutBoolean\(\)](#),
[tibasTuple_PutString\(\)](#), [tibasTuple_PutDateTime\(\)](#)[tibasTuple_PutBlob\(\)](#)

tibasTuple_PutString()

Function

Declaration `tibas_status` `tibasTuple_PutString`
 (`tibasTuple` `tuple`,
 `const char*` `fieldname`,
 `char*` `value`)

Purpose Stores a string value in a specified field in a specified tuple.

Parameter	Parameter	Description
	<code>tuple</code>	Specify the tuple into which you want to put a string value.
	<code>fieldname</code>	The fieldname.
	<code>value</code>	The string value to store in the fieldname.

Remarks Use the `tibasTuple_PutString()` function to put a specified string value in a specified field in a specified tuple.

See Also [tibasTuple_PutChar\(\)](#), [tibasTuple_PutShort\(\)](#), [tibasTuple_PutInt\(\)](#),
[tibasTuple_PutLong\(\)](#), [tibasTuple_PutFloat\(\)](#), [tibasTuple_PutDouble\(\)](#),
[tibasTuple_PutBoolean\(\)](#), [tibasTuple_PutDateTime\(\)](#)[tibasTuple_PutBlob\(\)](#)

tibasTuple_PutDateTime()

Function

Declaration `tibas_status` `tibasTuple_PutDateTime`
 (`tibasTuple` `tuple`,
 `const char*` `fieldName`,
 `tibas_DateTime*` `value`)

Purpose Stores a datetime value in a specified field in a specified tuple.

Parameter	Parameter	Description
	<code>tuple</code>	Specify the tuple into which you want to put a datetime value.
	<code>fieldName</code>	The fieldname.
	<code>value</code>	A <code>_tibasDateTime</code> struct that specifies the datetime value to store in the fieldname.

Remarks Use the `tibasTuple_PutDateTime()` function to put a specified datetime value in a specified field in a specified tuple.

The value field must specofy a `tibas_dateTime` structure. The `_tibasDateTime` struct is defined as follows:

```
struct _tibasDateTime {
    tibas_long sec;
    tibas_int nsec;
};
```

The members in the `_tibas_dateTime` struct are defined as follows:

- **sec** Specifies the time, in seconds, since the UNIX EPOCH time (1970-01-01 00:00:00 +0000).
- **nsec** Specifies the time, in nanoseconds, since the UNIX EPOCH time (1970-01-01 00:00:00 +0000). This field provides is extra timing information in addition to the number of seconds since the EPOCH time; usually this is set to 0.

Example The following example is taken from the `ASOperations.c` example program provided with the TIBCO ActiveSpaces distribution.

```
#include <time.h>

tibasTuple newValueTuple;
tibasTuple oldValueTuple;
tibasDateTime datetime;
```

```

time_t t;
tibas_status status;

printf("Put: Enter the key (integer): ");
gets(inputBuffer);
if (strcmp(inputBuffer, "") == 0) continue;
key = atoi(inputBuffer);
printf("Put: Enter the value (string): ");
    gets(inputBuffer);

/* Create a tuple and put the key into it. */
tibasTuple_Create(&newValueTuple);
tibasTuple_PutInt(newValueTuple, "key", key);

time(&t);
datetime.sec = (tibas_long) t;
datetime.nsec = 0;
tibasTuple_PutDateTime(newValueTuple, "time", datetime);

```

See Also [tibasTuple_PutChar\(\)](#), [tibasTuple_PutShort\(\)](#), [tibasTuple_PutInt\(\)](#),
[tibasTuple_PutLong\(\)](#), [tibasTuple_PutFloat\(\)](#), [tibasTuple_PutDouble\(\)](#),
[tibasTuple_PutString\(\)](#), [tibasTuple_PutBoolean\(\)](#), [tibasTuple_PutBlob\(\)](#),
[tibasTuple_GetDateTime\(\)](#)

tibasTuple_PutBlob()

Function

Declaration `tibas_status` `tibasTuple_PutBlob`
 `(tibasTuple` `tuple,`
 `const char*` `fieldname,`
 `char*` `value,`
 `tibas_int` `length)`

Purpose Stores a binary large object (BLOB) value in a specified field in a specified tuple.

Parameter	Parameter	Description
	<code>tuple</code>	Specify the tuple into which you want to put a boolean value.
	<code>fieldname</code>	The fieldname.
	<code>value</code>	A pointer to the start of the string of bytes making up the blob value.
	<code>length</code>	The length in number of bytes of the blob value.

Remarks Use the `tibasTuple_PutBlob()` function to put a specified blog value in a specified field in a specified tuple.

See Also [tibasTuple_PutChar\(\)](#), [tibasTuple_PutShort\(\)](#), [tibasTuple_PutInt\(\)](#),
[tibasTuple_PutLong\(\)](#), [tibasTuple_PutFloat\(\)](#), [tibasTuple_PutDouble\(\)](#),
[tibasTuple_PutString\(\)](#), [tibasTuple_PutBoolean\(\)](#)[tibasTuple_PutBoolean\(\)](#)

tibasTuple_Remove()

Function

Declaration `tibas_status` `tibasTuple_Remove`
 (`tibasTuple` `tuple`,
 const `char*` `fieldname`)

Purpose Removes a specified field from a specified tuple.

Parameter	Parameter	Description
	<code>tuple</code>	Specify the tuple for which you want to remove a field.
	<code>fieldname</code>	The fieldname for which to remove data.

Remarks Use the `tibasTuple_Remove()` function to remove the data in a specified field in a specified tuple.

See Also [tibasTuple_Clone\(\)](#)

tibasTuple_Size()

Function

Declaration `tibas_status` `tibasTuple_Size`
 (`tibasTuple` `tuple`,
 `tibas_int*` `size`)

Purpose Returns the number of fields contained in a specified tuple.

Parameter	Parameter	Description
	<code>tuple</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>size</code>	The address of a variable where the number of fields will be returned

Remarks Use the `tibasTuple_Size()` function to determine the number of fields in a specified tuple.

See Also [tibasTuple_IsNull\(\)](#)

tibasTuple_Clear()

Function

Declaration `tibas_status tibasTuple_Clear(
 tibasTuple tuple)`

Purpose Clears the data for the specified tuple.

Parameters	Parameter	Description
	tuple	The tuple on which to perform the action.

Remarks Use the `tibasTuple_Clear()` function to clear the data for a specified tuple.

See Also [tibasTuple_PutString\(\)](#)

tibasTuple_ToString()

Function

Declaration `tibas_status` `tibasTuple_ToString`
 `(tibasTuple tuple,`
 `char** tupleString)`

Purpose Creates a string representation of the specified tuple.

Parameter	Parameter	Description
	<code>tuple</code>	The TIBCO ActiveSpaces entity on which the function is invoked.
	<code>tupleString</code>	The string that is created and returned. Call <code>tibas_FreeData</code> to destroy the string.

Remarks Use the `tibasTuple_ToString()` function to convert the data in a specified tuple into a string that can be displayed or used by your application.

See Also [tibasTuple_Serialize\(\)](#), [tibasTuple_Free\(\)](#)

tibasTuple_Serialize()

Function

Declaration `tibas_status` `tibasTuple_Serialize`
 (`tibasTuple` `tuple`,
 `char**` `data`,
 `tibas_int*` `length`)

Purpose Serializes the tuple into a string of bytes.

Parameter	Parameter	Description
	<code>tuple</code>	The tuple on which the function is invoked.
	<code>data</code>	The serialized data that is created and returned. Call <code>tibas_FreeData</code> to destroy the data.
	<code>length</code>	The address of an <code>int</code> variable where the length in bytes of the serialized version of the tuple is returned.

Remarks Use the `tibasTuple_Serialize()` function to convert a specified tuple into a string of bytes.

See Also [tibasTuple_ToString\(\)](#), [tibasTuple_Deserialize\(\)](#)

tibasTuple_Deserialize()

Function

Declaration `tibas_status` `tibasTuple_Deserialize`
 (`tibasTuple*` `Tuple`,
 `char*` `data`,
 `tibas_int` `length`)

Purpose Creates a tuple object from a serialized form of a tuple.

Parameter	Parameter	Description
	<code>inTuple</code>	The tuple that is created and returned.
	<code>data</code>	The pointer to the serialized version of the tuple is located
	<code>length</code>	The length of the serialized version of the tuple

Remarks Use the `tibasTuple_Deserialize()` function to convert a tuple that has been serialized back into a deserialized format.

See Also [tibasTuple_Serialize\(\)](#)

tibasTuple_IsNull()

Function

Declaration `tibas_status` `tibasTuple_IsNull`
 (`tibasTuple` `tuple`,
 `const char*` `fieldname`)

Purpose Returns a boolean value indicating whether or not the specified tuple field is null (such as when it is not present in the tuple).

Parameter	Parameter	Description
	<code>tuple</code>	The tuple on which the function is invoked.
	<code>fieldname</code>	The fieldname used by the function.

Remarks Use the `tibasTuple_IsNull()` function to determine whether a specified tuple is NULL.

See Also [tibasTuple_Exists\(\)](#)

tibasTuple_Exists()

Function

Declaration `tibas_status` `tibasTuple_Exists`
 (`tibasTuple` `tuple`,
 `const char*` `fieldName`)

Purpose Returns a boolean value indicating whether the specified field exists.

Parameter	Parameter	Description
	<code>tuple</code>	The tuple on which the function is invoked.
	<code>fieldName</code>	The fieldname used by the function.

Remarks Use the `tibasTuple_Exists()` function to determine if a specified field in a specified tuple exists.

See Also [tibasTuple_IsNull\(\)](#)

tibasTuple_GetFieldNames()

Function

Declaration `tibas_status tibasTuple_GetFieldNames(
 tibasTuple tuple,
 tibasStringList* fieldNameList)`

Purpose Returns an array of names of fields inside the specified tuple.

Parameter	Parameter	Description
	tuple	The tuple on which the function is invoked.
	fieldNameList	A string list that is returned and which contains the list of field names contained in the tuple.

Remarks Use the `tibasTuple_GetFieldNames()` function to return a list of the field names in a specified tuple.

See Also [tibasTuple_GetFieldType\(\)](#)

tibasTuple_GetFieldType()

Function

Declaration `tibas_status` `tibasTuple_GetFieldType`
 (`tibasTuple` `tuple`,
 `tibas_type*` `fieldType`,
 `const char*` `fieldName`)

Purpose Returns the type of the specified field.

Parameter	Parameter	Description
	<code>tuple</code>	The tuple on which the function is invoked.
	<code>fieldType</code>	The type of the field returned by the function.
	<code>fieldNames</code>	The fieldnames used by the function.

Remarks Use the `tibasTuple_GetFieldType()` function to return the data type of a specified field in a specified tuple.

See Also [tibasTuple_GetFieldType\(\)](#)

tibasTuple_PutAll()

Function

Declaration `tibas_status tibasTuple_PutAll(
 tibasTuple tuple,
 tibasTuple otherTuple)`

Purpose Puts a collection of fields into the tuple.

Parameter	Parameter	Description
	tuple	The tuple in which the fields contained in <code>othertuple</code> will be put. If fields of the same name are contained in <code>otherTuple</code> , <code>tuple</code> ’s fields will be overwritten with the corresponding values of the fields in <code>otherTuple</code> .
	<code>otherTuple</code>	The tuple containing the fields that will be put into <code>tuple</code> .

Remarks Use the `tibasTuple_PutAll()` function to put a collection of fields into a specified tuple.

Merges the fields of the tuple specified as `otherTuple` into the tuple specified as `tuple`. If the field name of the incoming tuple matches the field inside the current tuple, the value of the field in the current tuple is overwritten with the value of the field from the incoming tuple.

See Also [tibasTuple_GetFieldNames\(\)](#)

tibasTuple_Eval()

Function

Declaration `tibas_status` `tibasTuple_Eval`
 (`tibasTuple` `tuple`,
 `tibas_boolean*` `result`,
 `char*` `filter`)

Purpose Evaluates if a specified tuple passes a specified filter.

Parameter	Parameter	Description
	<code>tuple</code>	The tuple on which the function is invoked.
	<code>result</code>	The result returned by the function:. Possible values TIBAS_TRUE if the tuple matches the filter and TIBAS_FALSE otherwise.
	<code>filter</code>	The filter string passed to the function.

Remarks Use the `tibasTuple_Eval()` function to determine whether a specified tuple passes a specified filter.

See Also [tibasTuple_GetFieldType\(\)](#)

tibasTuple_Free()

Function

```
Declaration      tibas_status tibasTuple_Free
                   (tibasTuple* tuple)
```

Purpose	Destroys allocated resources and frees memory.
----------------	--

Parameters

Parameter	Description
<code>tuple</code>	The TIBCO ActiveSpaces entity on which the function is invoked.

Remarks Use the `tibasTuple_Free()` function to free the memory and resources used by a specified tuple.

See Also [tibasTuple_Create\(\)](#)

Chapter 19 **TupleList**

This chapter explains the TupleList operations. TupleLists are lists of tuples that can be created and populated by your application.

Topics

- [TupleList Operations, page 384](#)

TupleList Operations

The following table lists the TupleList operations in alphabetical order:

Table 24 TupleList

Function or Type	Description	Page
tibasTupleList_Create()	Creates a tuple list.	385
tibasTupleList_Free()	Frees a specified tuple list.	389
tibasTupleList_Get()	Returns the tuple stored at position index in the list.	387
tibasTupleList_Put()	Appends a specified tuple to a specified tuple list.	388
tibasTupleList_Size()	Returns the number of items in a specified tuple list.	386

tibasTupleList_Create()

Function

Declaration

`tibas_status` titibasTupleList_Create(
tibasTupleList* tupleList)

Purpose

Creates a tuple list.

Parameters

Parameter	Description
tupleList	The tuple list created by the operation.

Remarks

Use the `tibasTupleList_Create()` function to create a tuple list.
A tuple list is a `tibasTupleList` object that is used to store a list of tuple objects.

See Also

`tibasTupleList_Free()`

tibasTupleList_Size()

Function

Declaration `tibas_status` `tibasTupleList_Size`(
 `tibasTupleList` `tupleList`,
 `tibas_int*` `size`)

Purpose Returns the number of items in a specified tuple list.

Parameters	Parameter	Description
	<code>tupleList</code>	Specify the tuple list for which you want to return the size.
	<code>size</code>	The size (in number of tuples) of the specified tuple list.

Remarks Use the `tibasTupleList_Size()` function to return the size of a specified tuple list.

See Also `tibasTupleList_Create()`, `tibasTupleList_Free()`

tibasTupleList_Get()

Function

Declaration

```
tibas_status tibasTupleList_Get(  
    tibasTupleList tupleList,  
    tibasTuple*     tuple,  
    tibas_int        index) )
```

Purpose

Returns the tuple stored at position index in the list.

Parameters

Parameter	Description
tupleList	Specify the tuple list form which you want to return a tuple.
tuple	The tuple returned by the operation.
index	The position index of the tuple to be returned by the operation.

Remarks

Use the `tibasTupleList_Get()` function to return a tuple that is at a specified index position in a specified tuple list.

Before you call `tibasTupleList_Get()`, call [tibasTupleList_Size\(\)](#) to determine the size of the list. The value that you specify with the `index` parameter should not be larger than the size of the list.

See Also

[tibasTupleList_Size\(\)](#)

tibasTupleList_Put()

Function

Declaration `tibas_status` `tibasTupleList_Put`(
 `tibasTupleList` `tupleList`,
 `tibasTuple` `tuple`)

Purpose Appends a specified tuple to a specified tuple list.

Parameters	Parameter	Description
	<code>tupleList</code>	Specify the name of the tuple list to which to append a tuple.
	<code>tuple</code>	The tuple to be appended to the tuple list.

Remarks Use the `tibasTupleList_Put()` function to append a tuple to a specified tuple list.

See Also [tibasTupleList_Get\(\)](#)

tibasTupleList_Free()

Function

Declaration `tibas_status tibasTupleList_Free(
 tibasTupleList* tupleList)`

Purpose Frees a specified tuple list.

Parameters	Parameter	Description
	<code>tupleList</code>	Specify the tuple list to free.

Remarks Use the `tibasTupleList_Free()` function to free the memory and resources used by a specified tuple list.

See Also [tibasTupleList_Create\(\)](#)

Chapter 20 **SpaceResult and InvokeResult**

This chapter explains the SpaceResult operations. You can use the SpaceResult operations to retrieve tuples from SpaceResult lists and perform other operations on a SpaceResult object.

A SpaceResult object contains a result of a batch space operation such as [tibasSpace_GetAll\(\)](#) and [tibasSpace_CompareAndPutAll\(\)](#).

SpaceResult objects are elements of a list of returned objects that is stored in a SpaceResultList object, which is described in [Chapter 21, SpaceResultList and InvokeResultList, on page 405](#). The SpaceResultList functions include additional convenience functions, such as [tibasSpaceResultList_GetStatus\(\)](#), which can be used to access the status code returned by an operation and the tuple associated with the operation.

InvokeResult objects contain the result of an invocation of an invocable function. You can use the InvokeResult operations to determine whether an invocation result has an error or determine the space member involved in the invocation.

Topics

- [SpaceResult and InvokeResult Operations, page 392](#)

SpaceResult and InvokeResult Operations

The following table lists the SpaceResult and InvokeResult operations:

Table 25 *SpaceResult and InvokeResult*

Function or Type	Description	Page
tibasSpaceResult_Free()	Frees a specified SpaceResult object.	397
tibasSpaceResult_GetTuple()	Gets the tuple associated with a specified SpaceResult, if one exists.	393
tibasSpaceResult_GetError()	Returns the error associated with a specified SpaceResult, if there is an error.	395
tibasSpaceResult_GetStatus()	Returns the status of the operation that generated the SpaceResult.	394
tibasSpaceResult_HasError()	Checks a specified SpaceResult object for errors.	396
tibasInvokeResult_GetStatus()	Returns the status of a specified invoke result.	398
tibasInvokeResult_GetReturn()	Returns the associated tuple value for a specified invoke result return.	399
tibasInvokeResult_GetMember()	Returns the member associated with a specified invoke result.	400
tibasInvokeResult_HasError()	Indicates whether a specified invoke result has an error.	401
tibasInvokeResult_GetError()	Returns errors contained in a specified invoke result object.	402
tibasInvokeResult_Free()	Frees a specified InvokeResult object.	403

tibasSpaceResult_GetTuple()

Function

Declaration `tibas_status tibasSpaceResult_GetTuple(
tibasSpaceResult result,
tibasTuple* tuple)`

Purpose Gets the tuple associated with a specified SpaceResult, if one exists.

Parameters	Parameter	Description
	result	The SpaceResult object on which the function is invoked.
	tuple	The tuple returned by the operation.

Remarks Use the `tibasSpaceResult_GetTuple()` function to return the tuple that is associated with a specified SpaceResult object.

See Also [tibasResultList_GetTuple\(\)](#), [tibasSpaceResultList_GetTuples\(\)](#)

tibasSpaceResult_GetStatus()

Function

Declaration `tibas_status` tibasSpaceResult_GetStatus(
 tibasSpaceResult result,
 tibas_status* status)

Purpose Returns the status of the operation that generated the SpaceResult.

Parameters	Parameter	Description
	result	The SpaceResult object on which the function is invoked.
	status	The status returned by the operation.

Remarks Use the `tibasSpaceResult_GetStatus()` function to check the status of a result from a space operation.

 The `result` parameter specifies the `SpaceResult` object for which to check the status, and the `status` parameter specifies the status.

See Also [tibasSpaceResultList_GetStatus\(\)](#)

tibasSpaceResult_GetError()

Function

Declaration

`tibas_status` `tibasSpaceResult_GetError`
`(tibasSpaceResult result,`
`tibasError *error)`

Purpose

Returns the error associated with a specified SpaceResult, if there is an error.

Parameters

Parameter	Description
result	The SpaceResult object on which the function is invoked.
error	If there is an error, returns a tibasError object.

Remarks

Use the `tibasSpaceResult_GetError()` function to return an error associated with a specified SpaceResult, if an error condition exists. Before calling `tibasSpaceResult_GetError()`, call the `tibasSpaceResult_HasError()` function to determine if there is an error.

See Also

`tibasSpaceResultList_GetError()`

tibasSpaceResult_HasError()

Function

Declaration `tibas_boolean` tibasSpaceResult_HasError(
 tibasSpaceResult result)

Purpose Checks a specified SpaceResult object for errors.

Parameters	Parameter	Description
	result	The SpaceResult object to be checked for errors.

Remarks Use the `tibasSpaceResult_HasError()` function to check a specified SpaceResult object for errors.

 If the space result has an error, the function returns `TIBAS_TRUE`. If there is no error, it returns `TIBAS_FALSE`.

 If the space result has an error, you can call the `tibasSpaceResult_GetError()` function to determine the error.

See Also `tibasSpaceResult_GetError()`

tibasSpaceResult_Free()

Function

Declaration [tibas_status](#) tibasSpaceResult_Free(tibasSpaceResult *result)

Purpose Frees a specified SpaceResult object.

Parameters	Parameter	Description
	result	The SpaceResult object on which the function is invoked.

Remarks Use the `tibasSpaceResult_Free()` function to free a specified SpaceResult object and the memory and resources associated with it.

See Also [tibasSpaceResultList_Free\(\)](#)

tibasInvokeResult_GetStatus()

Function

Declaration `tibas_status` `tibasInvokeResult_GetStatus(`
 `tibasInvokeResult` `result,`
 `tibas_status*` `status);`

Purpose Returns the status of a specified invoke result.

Parameters	Parameter	Description
	<code>result</code>	Specify the <code>InvokeResult</code> object for which to get a status.
	<code>status</code>	Returns the status of the invocation result.

Remarks Use the `tibasInvokeResult_GetStatus()` function to return the status of a call to an application function made using the `tibasSpace_Invoke()`, `tibasSpace_InvokeMember()`, `tibasSpace_InvokeMembers()`, `tibasSpace_InvokeRemoteMembers()`, or `tibasSpace_InvokeSeeders()` function.

If the status is OK, returns `TIBAS_OK`; otherwise, returns a status code indicating an error.

See Also [tibasSpace_Invoke\(\)](#), [tibasSpace_InvokeMember\(\)](#), [tibasSpace_InvokeMembers\(\)](#), [tibasSpace_InvokeRemoteMembers\(\)](#), [tibasSpace_InvokeSeeders\(\)](#)

tibasInvokeResult_GetReturn()

Function

Declaration `tibas_status tibasInvokeResult_GetReturn(
 tibasInvokeResult result,
 tibasTuple* tuple);`

Purpose Returns the associated tuple value for a specified invoke result return.

Parameters

Parameter	Description
result	Specify the InvokeResult object to return a tuple for.
tuple	Returns the result tuple.

Remarks Use the `tibasInvokeResult_GetReturn()` function to return the tuple that resulted from a call to an application function made using the `tibasSpace_Invoke()`, `tibasSpace_InvokeMember()`, `tibasSpace_InvokeMembers()`, `tibasSpace_InvokeRemoteMembers()`, or `tibasSpace_InvokeSeeders()` function.

If the tuple value is NULL, returns an error message.

See Also [tibasSpace_Invoke\(\)](#), [tibasSpace_InvokeMember\(\)](#), [tibasSpace_InvokeMembers\(\)](#), [tibasSpace_InvokeRemoteMembers\(\)](#), [tibasSpace_InvokeSeeders\(\)](#)

tibasInvokeResult_GetMember()

Function

Declaration `tibas_status tibasInvokeResult_GetMember(
 tibasInvokeResult result,
 tibasMember* member);`

Purpose Returns the member associated with a specified invoke result.

Parameters	Parameter	Description
	result	Specify the InvokeResult object to return a member for.
	member	Returns the member referenced by the application call.

Remarks Use the tibasInvokeResult_GetMember() function to return the space member referenced by an application function call made using the tibasSpace_Invoke(), tibasSpace_InvokeMember(),tibasSpace_InvokeMembers(), tibasSpace_InvokeRemoteMembers(), or tibasSpace_InvokeSeeders() function.

See Also [tibasSpace_Invoke\(\)](#), [tibasSpace_InvokeMember\(\)](#), [tibasSpace_InvokeMembers\(\)](#), [tibasSpace_InvokeRemoteMembers\(\)](#), [tibasSpace_InvokeSeeders\(\)](#)

tibasInvokeResult_HasError()

Function

Declaration `tibas_status tibasInvokeResult_HasError(
 tibasInvokeResult result);`

Purpose Indicates whether a specified invoke result has an error.

Parameters

Parameter	Description
result	Specify the InvokeResult object to examine for an error.

Remarks Use the `tibasInvokeResult_HasError()` function to determine whether a specified result from an application call to one of the invocation functions has an error.

If the result contains an error, the function returns `TIBAS_TRUE`. Otherwise it returns `TIBAS_FALSE`.

See Also [tibasSpace_Invoke\(\)](#), [tibasSpace_InvokeMember\(\)](#), [tibasSpace_InvokeMembers\(\)](#), [tibasSpace_InvokeRemoteMembers\(\)](#), [tibasSpace_InvokeSeeders\(\)](#)

tibasInvokeResult_GetError()

Function

Declaration `tibas_status tibasInvokeResult_GetError(
 tibasInvokeResult result,
 tibasError* error);`

Purpose Returns errors contained in a specified invoke result object.

Parameters	Parameter	Description
	result	Specify the InvokeResult object to return n error for.
	error	If there is an error, returns the error message for the error.

Remarks If the result of an application call to one of the invocation functions returns an error, use the `tibasInvokeResult_GetError()` function to return the error message associated with the error.

Before you call `tibasInvokeResult_GetError()`, call the `tibasInvokeResult_HasError()` function to determine whether the invocation result returned an error.

See Also `tibasInvokeResult_HasError()`, `tibasSpace_Invoke()`, `tibasSpace_InvokeMember()`, `tibasSpace_InvokeMembers()`, `tibasSpace_InvokeRemoteMembers()`, `tibasSpace_InvokeSeeders()`

tibasInvokeResult_Free()

Function

Declaration	tibas_status tibasInvokeResult_Free(tibasInvokeResult* result)	
Purpose	Frees a specified InvokeResult object.	
Parameters	Parameter	Description
	result	Specify the InvokeResult object to free.
Remarks	Use the tibasInvokeResult_Free() function to free the InvokeResult object returned by a call to one of the invocation functions.	
See Also	tibasSpace_Invoke(), tibasSpace_InvokeMember(), tibasSpace_InvokeMembers(), tibasSpace_InvokeRemoteMembers(), tibasSpace_InvokeSeeders()	

Chapter 21 **SpaceResultList and InvokeResultList**

This chapter explains the SpaceResultList and InvokeResultList operations. SpaceResultList objects are returned by batch operations. Besides giving access to individual Result objects, SpaceResultList objects provide convenience functions for accessing individual tuples and status values directly from the list.

InvokeResultList objects are returned by Invoke functions that return more than one InvokeResult object, such as [tibasSpace_InvokeMembers\(\)](#). Besides giving access to individual InvokeResult objects, InvokeResultList objects provide convenience functions for accessing individual tuples and status values directly from the list.

Topics

- [SpaceResultList and InvokeResultList Operations, page 406](#)

SpaceResultList and InvokeResultList Operations

The following table lists the SpaceResultList and InvokeResultList operations:

Table 26 *SpaceResultList and InvokeResultList*

Function or Type	Description	Page
tibasSpaceResultList_Free()	Frees a specified SpaceResult list.	416
tibasSpaceResultList_Get()	Returns the tuple stored at a specified position index in a specified SpaceResult list.	412
tibasSpaceResultList_GetTuples()	Returns a list of all the tuples that were returned in a specified resultList.	410
tibasResultList_GetTuple()	Returns the tuple stored at a specified position index in a specified SpaceResult list.	409
tibasSpaceResultList_GetError()	Returns the error stored in the SpaceResult at a specified index position in a specified SpaceResultList.	414
tibasSpaceResultList_GetStatus()	Returns the status value of the space result stored at a specified position in a specified SpaceResult list.	413
tibasSpaceResultList_HasError()	Checks whether a specified SpaceResultList contains a space result with an error.	408
tibasSpaceResultList_Put()	Appends a specified SpaceResult object to a specified SpaceResult list.	415
tibasSpaceResultList_Size()	Returns the number of tuples in the SpaceResultList object.	411
tibasInvokeResultList_Size()	Returns the number of tuples in the SpaceResultList object.	411
tibasInvokeResultList_Get()	Returns a tuple list containing the tuples returned in a specified invoke result list.	422
tibasSpaceResultList_GetStatus()	Returns the error stored in the InvokeResult at a specified index position in a specified InvokeResultList.	423

Table 26 *SpaceResultList and InvokeResultList*

Function or Type	Description	Page
tibasSpaceResultList_GetError()	Indicates whether an error exists in a specified invoke result list.	424
tibasInvokeResultList_Put()	Appends a specified invoke result to a specified invoke result list.	417
tibasInvokeResultList_Size()	Returns the size of a specified invoke result list.	418
tibasInvokeResultList_Get()	Returns the invoke result at a specified index position in a specified invoke result list.	419
tibasInvokeResultList_GetStatus()	Returns the status of an invoke result at a specified index position in a specified invoke result list.	420
tibasInvokeResultList_GetReturn()	Returns the tuple that was returned at a specified index position in a specified invoke result list.	421
tibasInvokeResultList_GetReturns()	Returns a tuple list containing the tuples returned in a specified invoke result list.	422
tibasInvokeResultList_GetError()	Returns the error stored in the InvokeResult at a specified index position in a specified InvokeResultList.	423
tibasInvokeResultList_HasError()	Indicates whether an error exists in a specified invoke result list.	424
tibasInvokeResultList_Free()	Frees a specified invoke result list.	425

tibasSpaceResultList_HasError()

Function

Declaration `tibas_boolean` tibasSpaceResultList_HasError(
 tibasSpaceResultList resultList)

Purpose Checks whether a specified SpaceResultList contains a space result with an error.

Parameters	Parameter	Description
	resultList	The SpaceResultList object on which the function is invoked.

Remarks Use the `tibasSpaceResultList_HasError()` function to determine whether any of the space results in a specified SpaceResult list has an error. If there is an error, the function returns `TIBAS_TRUE`.

 If the result list contains an error, you can call the `tibasSpaceResultList_GetError()` function to determine the error.

See Also `tibasSpaceResultList_GetError()`

tibasResultList_GetTuple()

Function

Declaration

```
tibas_status tibasSpaceResultList_GetTuple(  
    tibasSpaceResultList resultList,  
    tibasTuple*          tuple,  
    tibas_int             index)
```

Purpose

Returns the tuple stored at a specified position index in a specified SpaceResult list.

Parameters

Parameter	Description
resultList	The SpaceResultList object on which the function is invoked.
tuple	The tuple returned by the function.
index	The index position of the tuple to be returned.

Remarks

Use the `tibasSpaceResultList_GetTuple()` function to return a tuple value at a specified index position in a specified result list.

See Also

[tibasSpaceResultList_GetTuples\(\)](#), [tibasSpaceResult_GetTuple\(\)](#)

tibasSpaceResultList_GetTuples()

Function

Declaration `tibas_status tibasSpaceResultList_GetTuples(
 tibasSpaceResultList resultList,
 tibasTupleList* tupleList)`

Purpose Returns a list of all the tuples that were returned in a specified resultList.

Parameters

Parameter	Description
resultList	The SpaceResultList object on which the function is invoked.
tupleList	The list of tuples returned by the operation.

Remarks Use the tibasSpaceResultList_GetTuples() function to return a tupleList that contains the tuples that were returned in a specified resultList. A resultList object is returned by batch operations that operate on a space, such as [tibasSpace_TakeAll\(\)](#) or [tibasSpace_LockAll\(\)](#).

See Also [tibasResultList_GetTuple\(\)](#), [tibasSpaceResultList_GetStatus\(\)](#)

tibasSpaceResultList_Size()

Function

Declaration

```
tibas_status tibasSpaceResultList_Size(  
    tibasSpaceResultList resultList,  
    tibas_int* size)
```

Purpose

Returns the number of tuples in the SpaceResultList object.

Parameters

Parameter	Description
resultList	The SpaceResultList object on which the function is invoked.
size	The size (in number of tuple objects) of the specified SpaceResultList.

Remarks

Use the `tibasSpaceResultList_Size()` function to return the size of a specified SpaceResult list. Calling this function before you call the [tibasSpaceResultList_Get\(\)](#) function can prevent requesting a get of data from an index value higher than the upper limit of the list.

See Also

[tibasSpaceResultList_Get\(\)](#)

tibasSpaceResultList_Get()

Function

Declaration `tibas_status` `tibasSpaceResultList_Get`(
 `tibasSpaceResultList` `resultList`,
 `tibasSpaceResult*` `result`,
 `tibas_int` `index`)

Purpose Returns the space result stored at a specified position index in a specified SpaceResult list.

Parameters	Parameter	Description
	<code>resultList</code>	The SpaceResultList object on which the function is invoked.
	<code>result</code>	The SpaceResult object returned by the function.
	<code>index</code>	The position index of the SpaceResult object to be returned by the function.

Remarks Use the `tibasSpaceResultList_Get()` function to return the SpaceResult object at a specified index position in a specified SpaceResult list.

See Also [tibasResultList_GetTuple\(\)](#)

tibasSpaceResultList_GetStatus()

Function

Declaration `tibas_status` tibasSpaceResultList_GetStatus(
 tibasSpaceResultList resultList,
 tibas_status* status,
 tibas_int index)

Purpose Returns the status value of the space result stored at a specified position in a specified SpaceResult list.

Parameters	Description
resultList	The SpaceResultList object on which the function is invoked.
status	The status value returned by the function.
index	The index position of the SpaceResult object for which to return a status value.

Remarks Use the `tibasSpaceResultList_GetStatus()` function to return the status of a space result at a specified position in a SpaceResultList. If the status indicates that there is an error in the space result, you can call the `tibasSpaceResultList_GetError()` function to determine the error.

See Also `tibasSpaceResultList_GetError()`

tibasSpaceResultList_GetError()

Function

Declaration `tibas_status tibasSpaceResultList_GetError(
 tibasSpaceResultList resultList,
 tibasError* error,
 tibas_int index)`

Purpose Returns the error stored in the SpaceResult at a specified index position in a specified SpaceResultList.

Parameters	Parameter	Description
	resultList	The SpaceResultList object on which the function is invoked.
	error	The error returned by the function.
	index	The position index of the Error object to be returned.

Remarks Use the `tibasSpaceResultList_GetError()` function to determine the error condition that exist in a space result at a specified location in a SpaceResult list.

See Also [tibasSpaceResultList_GetStatus\(\)](#)

tibasSpaceResultList_Put()

Function

Declaration

```
tibas_status tibasSpaceResultList_Put(  
    tibasSpaceResultList resultList,  
    tibasSpaceResult      result)
```

Purpose

Appends a specified SpaceResult object to a specified SpaceResult list.

Parameters

Parameter	Description
resultList	The SpaceResultList object on which the function is invoked.
result	The SpaceResult object to be appended to the SpaceResult list.

Remarks

Use the `tibasSpaceResultList_Put()` function to append a specified space result object to a specified SpaceResult list.

See Also

[tibasSpaceResultList_Get\(\)](#)

tibasSpaceResultList_Free()

Function

Declaration `tibas_status tibasSpaceResult_Free(
 tibasSpaceResult* result)`

Purpose Frees a specified SpaceResult list.

Parameters	Parameter	Description
	resultList	The SpaceResultList object on which the function is invoked.

Remarks Use the `tibasSpaceResult_Free()` function to free the memory and resources associated with a specified SpaceResult list.

See Also [tibasSpaceResult_Free\(\)](#)

tibasInvokeResultList_Put()

Function

Declaration `tibas_status tibasInvokeResultList_Put(
 tibasInvokeResultList resultList,
 tibasInvokeResult result);`

Purpose Appends a specified invoke result to a specified invoke result list.

Parameters	Parameter	Description
	resultList	Specify the InvokeResult list to which to append a result.
	result	Specify the result to append to the list.

Remarks Use the `tibasInvokeResultList_Put()` function to append a specified invocation result to an invoke result list. Invoke result lists are returned by a call to the [tibasSpace_InvokeMembers\(\)](#), [tibasSpace_InvokeRemoteMembers\(\)](#), or [tibasSpace_InvokeSeeders\(\)](#) function.

See Also [tibasSpace_InvokeMembers\(\)](#), [tibasSpace_InvokeRemoteMembers\(\)](#), [tibasSpace_InvokeSeeders\(\)](#), [tibasSpaceResultList_Get\(\)](#)

tibasInvokeResultList_Size()

Function

Declaration `tibas_status tibasInvokeResultList_Size(
 tibasInvokeResultList resultList,
 tibas_int* size);`

Purpose Returns the size of a specified invoke result list.

Parameters	Parameter	Description
	resultList	Specifies the result list to return the size for.
	size	Returns the size of the list.

Remarks Use the `tibasInvokeResultList_Size()` function to determine the size of a specified result list. This is useful when coding iterative functions to loop through an invoke result list. Invoke result lists are returned by a call to the [tibasSpace_InvokeMembers\(\)](#), [tibasSpace_InvokeRemoteMembers\(\)](#), or [tibasSpace_InvokeSeeders\(\)](#) function.

See Also [tibasSpace_InvokeMembers\(\)](#), [tibasSpace_InvokeRemoteMembers\(\)](#), [tibasSpace_InvokeSeeders\(\)](#), [tibasSpaceResultList_Get\(\)](#)

tibasInvokeResultList_Get()

Function

Declaration

```
tibas_status tibasInvokeResultList_Get(  
    tibasInvokeResultList resultList,  
    tibasInvokeResult* result,  
    tibas_int index)
```

Purpose

Returns the invoke result at a specified index position in a specified invoke result list.

Parameters

Parameter	Description
resultList	Specifies the result list for which to return a result
result	Returns the result.
index	Specifies the index position for the get operation.

Remarks

Use the `tibasInvokeResultList_Get()` function to return the invocation result at a specified index position in an `invokeResult` list. Invoke result lists are returned by a call to the `tibasSpace_InvokeMembers()`, `tibasSpace_InvokeRemoteMembers()`, or `tibasSpace_InvokeSeeders()` function.

See Also

`tibasSpace_InvokeMembers()`, `tibasSpace_InvokeRemoteMembers()`, `tibasSpace_InvokeSeeders()`, `tibasInvokeResultList_Put()`

tibasInvokeResultList_GetStatus()

Function

Declaration `tibas_status` `tibasInvokeResultList_GetStatus(`
 `tibasInvokeResultList` `resultList,`
 `tibas_status*` `status,`
 `tibas_int` `index)`

Purpose Returns the status of an invoke result at a specified index position in a specified invoke result list.

Parameters	Parameter	Description
	<code>resultList</code>	Specifies the result list for which to get status.
	<code>status</code>	Returns the status for the result.
	<code>index</code>	Specifies the index position for which to return a status.

Remarks Use the `tibasInvokeResult_GetStatus()` function to return the status of a result from a call to an application function made using the `tibasSpace_InvokeMembers()`, `tibasSpace_InvokeRemoteMembers()`, or `tibasSpace_InvokeSeeders()` function. The function allows you to specify the index position in the `InvokeResult` list returned by these functions.

If the status is OK, returns `TIBAS_OK`; otherwise, returns a status code indicating an error.

See Also `tibasSpace_InvokeMembers()`, `tibasSpace_InvokeRemoteMembers()`, `tibasSpace_InvokeSeeders()`

tibasInvokeResultList_GetReturn()

Function

Declaration `tibas_status tibasInvokeResultList_GetReturn(
 tibasInvokeResultList resultList,
 tibasTuple* tuple,
 tibas_int index);`

Purpose Returns the tuple that was returned at a specified index position in a specified invoke result list.

Parameters

Parameter	Description
resultList	Specifies the resultList to get return values for.
tuple	Returns the tuple.
index	Specifies the index position to return a tuple for.

Remarks Use the `tibasInvokeResultList_GetReturn()` function to return the tuple that was returned at a specified position in a `InvokeResult` list. Invoke result lists are returned by a call to the [tibasSpace_InvokeMembers\(\)](#), [tibasSpace_InvokeRemoteMembers\(\)](#), or [tibasSpace_InvokeSeeders\(\)](#) function.

See Also [tibasSpace_InvokeMembers\(\)](#), [tibasSpace_InvokeRemoteMembers\(\)](#), [tibasSpace_InvokeSeeders\(\)](#)

tibasInvokeResultList_GetReturns()

Function

Declaration `tibas_status tibasInvokeResultList_GetReturns(
 tibasInvokeResultList resultList,
 tibasTupleList* tupleList);`

Purpose Returns a tuple list containing the tuples returned in a specified invoke result list.

Parameters	Parameter	Description
	resultList	Specifies the resultList to return the list of returns.
	tupleList	Returns a list of the returned tuples.

Remarks Use the `tibasInvokeResultList_GetReturns()` function to return a list of the tuples that were returned in a a specified InvokeResult list. Invoke result lists are returned by a call to the `tibasSpace_InvokeMembers()`, `tibasSpace_InvokeRemoteMembers()`, or `tibasSpace_InvokeSeeders()` function

See Also `tibasSpace_InvokeMembers()`, `tibasSpace_InvokeRemoteMembers()`, `tibasSpace_InvokeSeeders()`, `tibasInvokeResultList_Get()`

tibasInvokeResultList_GetError()

Function

Declaration `tibas_status tibasInvokeResultList_GetError(
 tibasInvokeResultList resultList,
 tibasError* error,
 tibas_int index)`

Purpose Returns the error stored in the InvokeResult at a specified index position in a specified InvokeResultList.

Parameters	Parameter	Description
	resultList	Specifies the resultList to check for an error.
	error	Returns the error at the specified index position.
	index	Specifies the index position to return the error for.

Remarks Use the `tibasInvokeResultList_GetError()` function to determine the error condition that exists in a invocation result at a specified location in an InvokeResult list.

See Also [tibasInvokeResultList_GetStatus\(\)](#), [tibasInvokeResultList_HasError\(\)](#)

tibasInvokeResultList_HasError()

Function

Declaration `tibas_status tibasInvokeResultList_HasError(
 tibasInvokeResultList resultList);`

Purpose Indicates whether an error exists in a specified invoke result list.

Parameters	Parameter	Description
	resultList	Specifies the resultList to check for errors.

Remarks Use the `tibasInvokeResultList_HasError()` function to check for an error in a specified invoke result list.

See Also [tibasSpace_InvokeMembers\(\)](#), [tibasSpace_InvokeRemoteMembers\(\)](#),
[tibasSpace_InvokeSeeders\(\)](#), [tibasInvokeResultList_GetStatus\(\)](#),
[tibasInvokeResultList_GetError\(\)](#)

tibasInvokeResultList_Free()

Function

Declaration `tibas_status tibasInvokeResultList_Free(
 tibasInvokeResultList* resultList);`

Purpose Frees a specified invoke result list.

Parameters	Parameter	Description
	resultList	Specifies the result list to free.

Remarks Use the tibasInvokeResultList_Free() function to free a specified InvokeResult list.

See Also [tibasSpace_InvokeMembers\(\)](#), [tibasSpace_InvokeRemoteMembers\(\)](#),
[tibasSpace_InvokeSeeders\(\)](#)

Chapter 22 **Filter**

This chapter explains the filter operations. The filter operations are used to create filters and apply them to specified tuples. Filters can be also be used in space browsers to limit the information that is browsed.

- [Filter Operations](#)

Filter Operations

The following table lists the Filter operations:

Table 27 Filter

Function or Type	Description	Page
tibasFilter_Create()	Returns a new filter object.	429
tibasFilter_Eval()	Evaluates a specified tuple based on a filter defined in a tibasFilter object.	430
tibasFilter_Free()	Destroys allocated resources and frees memory.	431

tibasFilter_Create()

Function

Declaration `tibas_status` tibasFilter_Create
 (`tibasFilter*` `filter`,
 `const char*` `filterString`)

Purpose Returns a new filter object.

Parameters

Parameter	Description
<code>filter</code>	The filter returned by the function.
<code>filterString</code>	The filter string used to create the filter.

Remarks Use the `tibasFilter_Create()` function to create a filter.

After you have created the filter, you can use it with the [tibasFilter_Eval\(\)](#) function to evaluate whether a specified tuple passes the filter. You can also apply the filter to a space browser that you create by calling the [tibasMetaspace_Browse\(\)](#) function.

See Also [tibasFilter_Eval\(\)](#), [tibasFilter_Free\(\)](#), [tibasMetaspace_Browse\(\)](#)

tibasFilter_Eval()

Function

Declaration `tibas_status` `tibasFilter_Eval`
 (`tibasFilter` `filter`,
 `tibasTuple` `tuple`,
 `tibas_boolean*` `result`)

Purpose Evaluates a specified tuple based on a filter defined in a `tibasFilter` object.

Parameters	Parameter	Description
	<code>filter</code>	Specify the <code>tibasFilter</code> object to use to evaluate the tuple.
	<code>tuple</code>	The tuple passed to the function.
	<code>result</code>	The status returned by the function.

Remarks Use the `tibasFilter_Eval()` function to evaluate a specified tuple using a specified `tibasFilter` object.

 If the tuple passes the filter, the function returns `TIBAS_TRUE`; otherwise it returns `TIBAS_FALSE`.

See Also [tibasFilter_Create\(\)](#)

tibasFilter_Free()

Function

```
Declaration      tibas_status tibasFilter_Free
                   (tibasFilter* filter)
```

Purpose	Destroys allocated resources and frees memory.
----------------	--

Parameters

Parameter	Description
<code>filter</code>	Specify the <code>tibasFilter</code> object that you want to free.

Remarks	Use the <code>tibasFilter_Free()</code> function to free the memory and resources associated with a specified filter.
----------------	---

See Also [tibasFilter_Create\(\)](#)

Chapter 23 **Member**

This chapter explains the Member operations. A Member represents either a metaspace member or a space member.

Topics

- [Member Operations](#)

Member Operations

The following table lists the Member operations:

Table 28 Member

Function or Type	Description	Page
tibasMember_Free()	Destroys allocated resources and frees memory for a specified member.	435
tibasMember_GetName()	Returns the globally unique name of the local member.	436
tibasMember_GetManagementRole()	Returns the role of the specified member of the metaspace.	437
tibasMember_GetDistributionRole()	Returns the distribution role of the member for the given space.	438
tibasMember_GetContext()	Returns context data that is associated with a specified space member.	439
tibasMember_GetHostAddress()	Returns the host address for a specified space member.	440
tibasMember_GetPort()	Returns the port used to communicate with a specified space member.	441
tibasMember_GetJoinTime()	Returns the time when a specified member joined the metaspace.	442
tibasMember_GetId()	Returns a member ID for a specified member name.	443

tibasMember_Free()

Function

Declaration `tibas_status` `tibasMember_Free`
 (`tibasMember*` `member`)

Purpose Destroys allocated resources and frees memory for a specified member.

Parameters	Parameter	Description
	<code>member</code>	Provide a <code>tibasMember</code> object that specifies which member to free.

Remarks Use the `tibasMember_Free()` function to free a specified space member.

See Also [tibasMetaspace_GetSelfMember\(\)](#)

tibasMember_GetName()

Function

Declaration `tibas_status` `tibasMember_GetName`
 (`tibasMember` `member`,
 `char**` `memberName`)

Purpose Returns the globally unique name of the local member.

Parameters	Parameter	Description
	<code>member</code>	Provide a <code>tibasMember</code> object that specifies which member to return the member name for.
	<code>memberName</code>	The member name returned by the function.

Remarks Use the `tibasMember_GetName()` function to return the member name for a specified `tibasMember` object.



The `memberName` data is owned by the `Member` object. There is no need to free it explicitly, as it will be freed when the `Member` object is freed.

See Also [tibasMetaspace_GetSelfMember\(\)](#)

tibasMember_GetManagementRole()

Function

Declaration `tibas_status tibasMember_GetManagementRole(
 tibasMember member,
 tibas_managementRole* managementRole)`

Purpose Returns the role of the specified member of the metaspace.

Parameters

Parameter	Description
member	Provide a tibasMember object that specifies which member to return the management role for.
role	Returns the management role. Possible values are TIBAS_ROLE_NONE, TIBAS_ROLE_MEMBER, or TIBAS_ROLE_MANAGER

Remarks Use the `tibasMember_GetManagementRole()` function to return the management role that is specified for a specified space member.

The management role is assigned by an internal ActiveSpaces protocol that manages group membership. At a given time, a member can have one of the following management roles:

- **TIBAS_ROLE_NONE** A transitory role that is in effect while the group membership is being processed.
- **TIBAS_ROLE_MEMBER** The member is simply a member of the metaspace and is not managing membership.
- **TIBAS_ROLE_MANAGER** The member is currently managing membership for the Metaspace.

Your application might use this function to determine the current management role for a member that is intended to serve some special function in your application that would require that it have a management role or not have a management role.

See Also [tibasMember_GetDistributionRole\(\)](#)

tibasMember_GetDistributionRole()

Function

Declaration `tibas_status` `tibasMember_GetDistributionRole`(
 `tibasMember` `member`,
 `char*` `spaceName`,
 `tibas_distributionRole*` `distributionRole`)

Purpose Returns the distribution role of the member for the given space.

Parameters	Parameter	Description
	<code>member</code>	Provide a <code>tibasMember</code> object that specifies which member to return the distribution role for.
	<code>spaceName</code>	The name of the space for which the member's distribution role will be returned.
	<code>distributionRole</code>	<ul style="list-style-type: none">• The distribution role of the member. Can be one of the following:• <code>TIBAS_DISTRIBUTION_ROLE_NONE</code>• <code>TIBAS_DISTRIBUTION_ROLE_LEECH</code>• <code>TIBAS_DISTRIBUTION_ROLE_SEEDER</code>

Remarks Use the `tibasMember_GetDistributionRole()` function to return the distribution role that is assigned to a specified space member. The distribution role can be one of the following:

- **TIBAS_DISTRIBUTION_ROLE_SEEDER** Specifies that the space member is a seeder—an application that participates in the storing of data in the space, and can read and write data. When seeder applications join or leave the space, ActiveSpaces redistributes the data in the space as necessary to maintain even data distribution.
- **TIBAS_DISTRIBUTION_ROLE_LEECH** Specifies that the space member is a leech—an application that passively participates in the space and does not read and write data or cause redistribution of space data when it joins or leaves the space.
- **TIBAS_DISTRIBUTION_ROLE_NONE** Indicates that the specified member has not joined the space.

See Also [tibasMetaspace_GetSelfMember\(\)](#),

tibasMember_GetContext()

Function

Declaration `tibas_status` `tibasMember_GetContext`(
 `tibasMember` `member`,
 `tibasTuple*` `member`)

Purpose Returns context data that is associated with a specified space member.

Parameters	Parameter	Description
	<code>member</code>	Specify the member name for a member for which you want to return context data.
	<code>context</code>	Returns the context information associated with the member.

Remarks Use the `tibasMember_GetContext()` function to return context information associated with a specified member. For example, the `memberDef` for a specified member might have been set up to include an application name.

See Also [tibasMemberDef_SetContext\(\)](#)

tibasMember_GetHostAddress()

Function

Declaration `tibas_status` `tibasMember_GetHostAddress`(
 `tibasMember` `c_member`,
 `char**` `hostAddress`)

Purpose Returns the host address for a specified space member.

Parameters	Parameter	Description
	<code>c_member</code>	Specify the member for which you want to return the host address.
	<code>hostAddress</code>	Returns the host address of the specified member.

Remarks Use the `tibasMember_GetHostAddress()` function to return the host address for the device that is running a specified member.

See Also [tibasMember_GetPort\(\)](#), [tibasMember_GetJoinTime\(\)](#)

tibasMember_GetPort()

Function

Declaration `tibas_status` tibasMember_GetPort(
 tibasMember c_member,
 int* port)

Purpose Returns the port used to communicate with a specified space member.

Parameters

Parameter	Description
c_member	Specify the member definition for the member for which you want to query.
port	Returns the port used by the member.

Remarks Use the `tibasMember_GetPort()` function to return the port that is used by a specified member.

See Also [tibasMember_GetHostAddress\(\)](#), [tibasMember_GetJoinTime\(\)](#)

tibasMember_GetJoinTime()

Function

Declaration `tibas_status` tibasMember_GetJoinTime(
 `tibasMember` c_member,
 `time_t*` joinTime)

Purpose Returns the time when a specified member joined the metaspace.

Parameters	Parameter	Description
	c_member	Specify the memberDef for the member for which you want to return the join time.
	joinTime	Returns the time when the member joined the metaspace.

Remarks Use the `tibasMember_GetJoinTime()` function to return the time when a specified member joined the metaspace.

See Also [tibasMember_GetHostAddress\(\)](#), [tibasMember_GetPort\(\)](#)

tibasMember_GetId()

Function

Declaration `tibas_status` tibasMember_GetId(
 tibasMember member,
 char* memberId);

Purpose Returns a member ID for a specified member name.

Parameters	Parameter	Description
	member	Specify the member name for a member for which you want to return a member ID.
	memberId	Returns a member ID for a specified member name.

Remarks Use the tibasMember_GetId() function to return a member ID for a specified member. You can then use the member ID as a means to keep track of the member.

See Also [tibasMember_GetName\(\)](#)

Chapter 24 **MemberList**

This chapter explains the `MemberList` operations.

Topics

- [MemberList Operations, page 446](#)

MemberList Operations

The following table lists the MemberList operations in alphabetical order:

Table 29 MemberList

Function or Type	Description	Page
tibasMemberList_Free()	Frees the memory and resources used by a specified tibasMemberList object.	449
tibasMemberList_Get()	Returns the member object stored at the specified position index in a specified MemberList object.	448
tibasMemberList_Size()	Returns the number of entries in a specified MemberList object.	447

tibasMemberList_Size()

Function

Declaration `tibas_status` tibasMemberList_Size
 (`tibasMemberList` memberList,
 `tibas_int` *size)

Purpose Returns the number of entries in a specified MemberList object.

Parameters	Parameter	Description
	memberList	The MemberList object on which the function is invoked.
	size	The number of members in the specified member list.

Remarks Use the `tibasMemberList_Size()` function to determine the number of entries that are in a specified memberList object.

 You can call this function before calling the `tibasMemberList_Get()` function, to ensure that index position for which you are trying to get a Member object is within the range of entries in the Member List.

 Use the `tibasMetaspace_GetSpaceMembers()` function to return a `tibasMemberList` object that lists the space members for a specified space.

See Also `tibasMetaspace_GetSpaceMembers()`

tibasMemberList_Get()

Function

Declaration `tibas_status` `tibasMemberList_Get`
 (`tibasMemberList` `memberList`,
 `tibasMember` *`member`,
 `tibas_int` `index`)

Purpose Returns the member object stored at the specified position `index` in a specified `MemberList` object.

Parameters	Parameter	Description
	<code>memberList</code>	The <code>MemberList</code> object on which the function is invoked.
	<code>member</code>	The <code>Member</code> object returned by the function.
	<code>index</code>	The index position of the member object to be returned by the function.

Remarks Use the `tibasMemberList_Get()` function to return the member object at a specified index position in a specified Member List.

 Use the `tibasMetaspace_GetSpaceMembers()` function to obtain a `tibasMemberList` object to pass to `tibasMemberListGet()`.

See Also `tibasMetaspace_GetSpaceMembers()`

tibasMemberList_Free()

Function

Declaration	<code>tibas_status</code> tibasMemberList_Free (tibasMemberList *memberList)	
Purpose	Frees the memory and resources used by a specified tibasMemberList object.	
Parameters	Parameter	Description
	memberList	Specify the tibasMemberList object to free.
Remarks	Use the <code>tibasMemberList_Free()</code> function to free the memory and resources used by a specified Member List.	
See Also	tibasMember_Free()	

Chapter 25

Persister, Action, Op, OpList, ActionResult, LogLevel, and Recovery Functions

This chapter explains the Persister, Action, Op, OpList, ActionResult, LogLevel, and Recovery operations.



the information on persister functions in this chapter pertains only to shared-all persistence. If you configure shared-nothing persistence, ActiveSpaces manages persistence internally.

The Persister object enables the user to specify a set of functions that collectively implement the Persister interface. Once a Persister object is created, it can then be registered on a space using the `tibasSpace_SetPersister` function.

The Action, Op, OpList, and ActionResult classes are closely related to the Persister class, and hence all are included in this chapter. Functions implementing persistence get passed a `tibasAction` and return a `tibasActionResult`. The `tibasAction` objects have a type, contain the space for which the action applies and its name. Actions of type `TIBAS_ACTION_READ` also contain a tuple (the values of the key fields for the record that needs to be read from the persistence layer) and Actions of type `TIBAS_ACTION_WRITE` contain an `OpList`. `Op` objects represent individual persistence operations and have a type.

Recovery options enable data recovery in the metaspace.

Topics

- [Persister Operations, page 452](#)
- [Action, Op, OpList, ActionResult, Invocable, and LogFile Operations, page 456](#)

Persister Operations

The following table lists the Persister operations in alphabetical order:

Table 30 Persister

Function or Type	Description	Page
tibasPersister_Create()	Creates a Persister object that can be registered on a space. The arguments of this call are used to specify the names of functions that will be invoked at various points in the life cycle of a space.	453
tibasPersister_Free()	Destroys allocated resources and frees memory.	455

tibasPersister_Create()

Function

Declaration

```
tibas_status tibasPersister_Create
(tibasPersister *persister,
 tibas_onAction onOpen,
 tibas_onAction onLoad,
 tibas_onAction onWrite,
 tibas_onAction onClose,
 tibas_onAction onRead,
 tibas_onAction onAlter)
```

Purpose

Creates a Persister object that can be registered on a space. The arguments of this call are used to specify the names of functions that will be invoked at various points in the life cycle of a space.

Parameters

Parameter	Description
persister	The Persister object created by the operation.
onOpen	Specifies the function to be invoked when the Persister is registered.
onLoad	Specifies the function to be invoked when the space is in its load state.
onWrite	Specifies the function to be invoked when the space is in its ready state.
onRead	Specifies function to be invoked when the space is in its ready state.
onClose	Specifies the function to be invoked when the members leave the Metaspace or tibasSpace_StopPersister is invoked.
onAlter	Specifies the function to be invoked when the space definition is altered.

Remarks

If you are implementing shared all persistence, use the `tibasPersister_Create()` function to create a Persister object that you can use to persist application data.

The persister object that you create specifies functions that will be called at various points in the life cycle of the space; for example, when the space is in the READY state to read data from local storage, or when a member leaves the space, to write data to local storage.

After you have created the persister object, use the `tibasSpace_SetPersister()` function to register the persister for a specified space.

Before you can call `tibasSpace_SetPersister()`, you must create a persister object by calling the `tibasPersister_Create()` function.

The name that you assign with the persister parameter specifies the name of a function that your application provides to call the functions that are called at specified points in the life cycle of the space, to implement persistence.

See Also `tibasSpaceDef_SetPersisted()`, `tibasSpace_SetPersister()`, `tibasSpace_SetPersister()`

tibasPersister_Free()

Function

Declaration `tibas_status` tibasPersister_Free(tibasPersister *persister)

Purpose Destroys allocated resources and frees memory.

Parameters	Parameter	Description
	persister	The Persister object on which the operation is invoked.

Remarks Use the `tibasPersister_Free()` function to free a specified Persister object and its associated memory and resources.

See Also [tibasPersister_Create\(\)](#)

Action, Op, OpList, ActionResult, Invocable, and LogFile Operations

The following table lists the Action, Op, OpList, ActionResult, LogFile, and Invocable operations:

Table 31 Action, Op, OpList, ActionResult, Invocable, and LogFile

Function or Type	Description	Page
Action Operations		
tibasAction_GetType()	Returns the action type for a specified action.	459
tibasAction_GetTuple()	For actions of the type TIBAS_ACTION_READ, returns a tuple containing the key fields for the tuple that needs to be retrieved from the persistence layer	460
tibasAction_GetSpace()	Returns a valid Space object for the space that triggered this invocation of the persister function. Applicable to Actions of all types.	461
tibasAction_GetSpaceName()	Returns the name of the space that triggered the invocation of the persister function. Applicable to Actions of all types.	462
tibasAction_GetOps()	For actions of the type TIBAS_ACTION_WRITE, returns a list of operations to apply to the persistence layer.	463
tibasAction_Free()	Frees a specified action object.	464
Op Operations		
tibasOp_GetType()	Returns the type of a specified persistence operation.	465
tibasOp_GetTuple()	Returns the tuple associated with a specified Op. Applicable to all types of Op.	466

Table 31 Action, Op, OpList, ActionResult, Invocable, and LogFile

Function or Type	Description	Page
tibasOp_GetOldTuple()	For Ops of the type TIBAS_OP_PUT, returns the oldTuple that was overwritten (updated) by the put if there was one, or NULL (if the put was an insert).	467
tibasOp_HasOldTuple()	For applicable to Ops of the type TIBAS_OP_PUT, indicates whether the put operation overwrote an existing tuple.	468
tibasOp_Free()	Frees a specified Op object.	469
OpList Operations		
tibasOpList_Size()	Returns the number of ops in a specified OpList object.	470
tibasOpList_Get()	Returns the Op stored at a specified position index in an Op list.	471
tibasOpList_Free()	Frees a specified OpList.	472
ActionResult Operations		
tibasActionResult_SetFailed()	Sets the state of the ActionResult to failed.	473
tibasActionResult_GetStatus()	Reserved for future use.	474
tibasActionResult_SetTuple()	Stores a tuple retrieved from persisted storage into an action result object.	475
tibasActionResult_GetTuple()	Reserved for future use.	476
Invocable Operations		
tibas_SetInvocable()	Specifies the name of an application function to be invoked.	477
tibas_SetMemberInvocable()	Specifies the name of an application function to be invoked on a specified space member.	479
LogLevel Operations		

Table 31 *Action, Op, OpList, ActionResult, Invocable, and LogFile*

Function or Type	Description	Page
tibas_SetLogLevel()	Sets the log level used by ActiveSpaces.	481
tibas_GetLogLevel()	Returns the log level used by ActiveSpaces.	482
tibas_EnableFileLogging()	Enables logging of specified log level messages to a specified log file.	483
tibas_DisableFileLogging()	Disables file logging.	484

tibasAction_GetType()

Function

```
Declaration      tibas_status tibasAction_GetType
                    (tibasAction      action,
                     tibas_actionType* actionType)
```

Purpose	Returns the action type for a specified action.
----------------	---

Parameters

Parameter	Description
action	The Action object on which the operation is performed.
actionType	Where the Action's type will be returned.

Remarks	Use the <code>tibasAction_GetType()</code> function to determine the action type for a specified action. Your persister function can use the action type to determine what steps to take to persist data.
----------------	---

The action type can be one of the following:

- **TIBAS_ACTION_OPEN** Indicates that a persister object is registered with the space.
- **TIBAS_ACTION_LOAD** Indicates that a persister object is registered with a space
- **TIBAS_ACTION_WRITE** Indicates that a tuple stored in the space is modified (due to a put, take, or update operation).
- **TIBAS_ACTION_READ** Indicates that the space has a capacity set and a capacity policy of EVICT, and a request to read, lock, or take a tuple by key value did not result in a matching tuple being found in the space.
- **TIBAS_ACTION_CLOSE** Indicates that a persister object is stopped for a space.

See Also [tibasAction_GetOps\(\)](#)[tibasActionResult_GetStatus\(\)](#)

tibasAction_GetTuple()

Function

Declaration `tibas_status` `tibasAction_GetTuple`
 (`tibasAction` `action`,
 `tibasTuple*` `tuple`)

Purpose For actions of the type `TIBAS_ACTION_READ`, returns a tuple containing the key fields for the tuple that needs to be retrieved from the persistence layer

Parameters	Parameter	Description
	<code>action</code>	The Action object on which the operation is performed.
	<code>tuple</code>	Where the tuple will be returned.

Remarks When processing actions of the type `TIBAS_ACTION_READ`, use the `tibasAction_GetTuple()` function to return a tuple containing the key fields for the tuple that needs to be retrieved from the persistence layer.

See Also [tibasAction_GetOps\(\)](#)

tibasAction_GetSpace()

Function

```
Declaration      tibas_status tibasAction_GetSpace
                   (tibasAction action,
                    tibasSpace* space)
```

Purpose	Returns a valid Space object for the space that triggered this invocation of the persist function. Applicable to Actions of all types.
----------------	--

Parameters

Parameter	Description
action	The Action object on which the operation is performed.
space	Returns the a Space object associated with the action.

Remarks	Use the <code>tibasAction_GetSpace()</code> function to return the Space object associated with a specified action.
----------------	---

See Also [tibasAction_GetType\(\)](#), [tibasAction_GetTuple\(\)](#), [tibasAction_GetSpaceName\(\)](#), [tibasAction_GetOps\(\)](#)

tibasAction_GetSpaceName()

Function

Declaration `tibas_status` `tibasAction_GetSpaceName`(
 `tibasAction` `action`,
 `char**` `spaceName`)

Purpose Returns the name of the space that triggered the invocation of the persister function. Applicable to Actions of all types.

Parameters	Parameter	Description
	<code>action</code>	The Action object on which the operation is performed.
	<code>spaceName</code>	Returns the name of the space associated with the action.

Remarks Use the `tibasAction_GetSpaceName()` function to return the space name for the Space object associated with a specified action.

See Also [tibasAction_GetType\(\)](#), [tibasAction_GetTuple\(\)](#), [tibasAction_GetSpace\(\)](#), [tibasAction_GetOps\(\)](#)

tibasAction_GetOps()

Function

Declaration	<code>tibas_status</code> <code>tibasAction_GetOps</code> (<code>tibasAction</code> <code>action</code> , <code>tibasOpList*</code> <code>opList</code>)							
Purpose	For actions of the type <code>TIBAS_ACTION_WRITE</code> , returns a list of operations to apply to the persistence layer.							
Parameters	<table><tr><th>Parameter</th><th>Description</th></tr><tr><td><code>action</code></td><td>Specify the action object for which to return a list of operations.</td></tr><tr><td><code>opList</code></td><td>Returns a list of operations to be carried out.</td></tr></table>		Parameter	Description	<code>action</code>	Specify the action object for which to return a list of operations.	<code>opList</code>	Returns a list of operations to be carried out.
Parameter	Description							
<code>action</code>	Specify the action object for which to return a list of operations.							
<code>opList</code>	Returns a list of operations to be carried out.							
Remarks	<p>Use the <code>tibasAction_GetOps()</code> function to return a list of the operations to be carried out for a specified action.</p> <p>Use this function in code that your application provides to implement shared all persistence.</p>							
See Also	<code>tibasAction_GetType()</code> , <code>tibasAction_GetTuple()</code> , <code>tibasAction_GetSpace()</code> , <code>tibasAction_GetSpaceName()</code>							

tibasAction_Free()

Function

Declaration `tibas_status` `tibasAction_Free`
 (`tibasAction*` `action`)

Purpose Frees a specified action object.

Parameters	Parameter	Description
	<code>action</code>	Specify the Action object to be freed.

Remarks Use the `tibasAction_Free()` function to free the memory used for a specified action object.

 Call this function at the end of the a function implementing the persister action, to avoid memory leaks.

See Also [tibasAction_GetType\(\)](#), [tibasAction_GetTuple\(\)](#), [tibasAction_GetSpace\(\)](#),
 [tibasAction_GetSpaceName\(\)](#)

tibasOp_GetType()

Function

Declaration `tibas_status` `tibasOp_GetType`
 (`tibasOp` `op`,
 `tibas_opType*` `opType`)

Purpose Returns the type of a specified persistence operation.

Parameters	Parameter	Description
	<code>op</code>	The Op object on which the operation is performed.
	<code>opType</code>	Where the type of the Op is returned.

Remarks Use the `tibasOp_GetType()` function to return the type of a specified persistence operation. The operation type can be one of the following:

- **TIBAS_OP_PUT** The operation is a Put operation.
- **TIBAS_OP_TAKE** The operation is a Take operation.

See Also [tibasOp_GetTuple\(\)](#), [tibasOp_GetOldTuple\(\)](#), [tibasOp_HasOldTuple\(\)](#),
[tibasOp_Free\(\)](#)

tibasOp_GetTuple()

Function

Declaration

```
tibas_status tibasOp_GetTuple(  
    tibasOp      op,  
    tibasTuple* tuple);
```

Purpose

Returns the tuple associated with a specified Op. Applicable to all types of Op.

Parameters

Parameter	Description
op	Specify the Op object for which to get the associated tuple.which the operation is performed.
tuple	Returns the tuple associated with the Op.

Remarks

Use the `tibasOp_GetTuple()` function to return the tuple associated with a specified operation.

See Also

[tibasOp_GetTuple\(\)](#), [tibasOp_GetOldTuple\(\)](#), [tibasOp_HasOldTuple\(\)](#), [tibasOp_Free\(\)](#)

tibasOp_GetOldTuple()

Function

Declaration `tibas_status` `tibasOp_GetOldTuple`
 (`tibasOp` `op`,
 `tibasTuple*` `oldTuple`)

Purpose For Ops of the type TIBAS_OP_PUT, returns the oldTuple that was overwritten (updated) by the put if there was one, or NULL (if the put was an insert).

Parameters	Parameter	Description
	<code>op</code>	The Op object on which the operation is performed.
	<code>oldTuple</code>	Where the old tuple that was overwritten (updated) by the put is returned.

Remarks Use the `tibasOp_GetOldTuple()` function to return the value that was written by a Put operation. You can use this function in application code to process a TIBAS_OP_PUT.

See Also [tibasAction_GetOps\(\)](#), [tibasOp_GetType\(\)](#), [tibasOp_HasOldTuple\(\)](#)

tibasOp_HasOldTuple()

Function

Declaration	<code>tibas_status</code> <code>tibasOp_HasOldTuple</code> (<code>tibasOp op</code>)				
Purpose	For applicable to Ops of the type <code>TIBAS_OP_PUT</code> , indicates whether the put operation overwrote and existing tuple.				
Parameters	<table><tr><th>Parameter</th><th>Description</th></tr><tr><td><code>op</code></td><td>The Op object on which the operation is performed.</td></tr></table>	Parameter	Description	<code>op</code>	The Op object on which the operation is performed.
Parameter	Description				
<code>op</code>	The Op object on which the operation is performed.				
Remarks	<p>Use <code>tibasOp_HasOldTuple()</code> function to determine whether a <code>TIBAS_OP_PUT</code> operation overwrote and existing tuple.</p> <p>The function returns <code>TIBAS_TRUE</code> if the put that triggered this Op overwrote (was an update) an existing tuple in the space or <code>TIBAS_FALSE</code> if the put that triggered this Op did not overwrite an existing tuple in the space (was an insert).</p>				
See Also	<code>tibasOp_GetOldTuple()</code> , <code>tibasOp_GetType()</code>				

tibasOp_Free()

Function

Declaration `tibas_status` tibasOp_Free
 (`tibasOp*` op)

Purpose Frees a specified Op object.

Parameters

Parameter	Description
op	The Op object on which the operation is performed.

Remarks Use the `tibasOp_Free()` function to free a specified Op object.

 `tibasOp_Free()` should be called by application code that implements shared all persistence after dta hs been persisted, to avoid memory leaks.

See Also [tibasAction_Free\(\)](#),

tibasOpList_Size()

Function

Declaration `tibas_status tibasOpList_Size`
 (`tibasOpList opList, tibas_int* size`)

Purpose Returns the number of ops in a specified OpList object.

Parameters	Parameter	Description
	opList	The OpList object on which the function is invoked.
	size	The number of ops in the specified op list.

Remarks Use the `tibasOpList_Size()` function to return the size of a specified OpList.

See Also [tibasAction_GetOps\(\)](#),

tibasOpList_Get()

Function

Declaration `tibas_status` tibasOpList_Get
 (`tibasOpList` `opList`,
 `tibasOp*` `op`,
 `tibas_int` `index`)

Purpose Returns the Op stored at a specified position index in an Op list.

Parameters	Parameter	Description
	<code>opList</code>	The OpList object on which the function is invoked.
	<code>op</code>	The Op object returned by the function.
	<code>index</code>	The position index of the Op object to be returned by the function.

Remarks Use the `tibasOpList_Get()` function to return the operation that is a a specified index position in a specified opList.

See Also [tibasAction_GetOps\(\)](#), [tibasOpList_Size\(\)](#)

tibasOpList_Free()

Function

Declaration `tibas_status tibasOpList_Free
(tibasOpList* opList)`

Purpose Frees a specified OpList.

Parameters	Parameter	Description
	opList	The OpList object on which the function is invoked.

Remarks Use the `tibasOpList_Free()` function to free a specified OpList.
Should be invoked at the end of a function implementing shared all persistence, to avoid memory leaks.

See Also [tibasOpList_Get\(\)](#)

tibasActionResult_SetFailed()

Function

Declaration `tibas_status tibasActionResult_SetFailed
(tibasActionResult result)`

Purpose Sets the state of the ActionResult to failed.

Parameters

Parameter	Description
result	The ActionResult object on which the operation is performed.

Remarks Use the tibasActionResult_SetFailed() function to set the state of a specified ActionResult object to FAILED.

See Also [tibasActionResult_GetStatus\(\)](#)

tibasActionResult_GetStatus()

Function

```
Declaration    tibas_status tibasActionResult_GetStatus
                 (tibasActionResult result, tibas_status* status)
```

Purpose	Reserved for future use.
----------------	--------------------------

Parameters

Parameter	Description
result	The ActionResult object on which the operation is performed.
status	The status returned by the function.

Remarks	This function is reserved for future use.
----------------	---

See Also [tibasActionResult_SetFailed\(\)](#)

tibasActionResult_GetTuple()

Function

Declaration

`tibas_status` `tibasActionResult_GetTuple`
`(tibasActionResult result,`
`tibasTuple* tuple)`

Purpose

Reserved for future use.

Parameters

Parameter	Description
<code>result</code>	The <code>ActionResult</code> object on which the operation is performed.
<code>tuple</code>	The tuple returned by the function.

Remarks

This function is reserved for future use.

See Also

`tibasActionResult_SetTuple()`

tibas_SetInvocable()

Function

Declaration `tibas_status` `tibas_SetInvocable`(
 `const char*` `name`,
 `tibas_invocable` `invocable`);

Purpose Specifies the name of an application function to be invoked.

Parameters	Parameter	Description
	<code>name</code>	Specifies the name of an application function to be invoked.
	<code>invocable</code>	Specifies a <code>tibas_invocable</code> structure that sets the space, tuple, and context for the invocation, and also returns the result of the function call.

Remarks Use the `tibas_SetInvocable()` function to specify a function that is called on a seeder. You must call `tibas_SetInvocable()` before you call the `tibasSpace_Invoke()` function.

The `name` parameter specifies the name of the invocable function that is called.

The `invocable` parameter specifies a `tibas_invocable` type that provides the name of a function that conforms to the function prototype for invocable functions.

The `tibas_invocable` typedef specifies the function prototype for an invocable function, and has the following syntax:

```
typedef void (TIBAS_CALL_API *tibas_invocable)(
    tibasSpace space,
    tibasTuple tuple,
    tibasTuple context,
    tibasTuple result);
```

where:

- **space** Specifies the space on which the invocable function is to be invoked.
- **tuple** Specifies a tuple that will be used to determine which entry to invoke the function on.
- **context** Specifies a user-defined context string used to identify the function.
- **result** Returns the result of the invocation.

For a sample program showing the use of `tibas_SetInvocable()` and `tibas_SpaceInvoke()`, see the **InvokeClient** sample program provided with the ActiveSpaces distribution. For documentation on `InvokeClient`, see [Remote Space Invocation: InvokeClient](#) in the *TIBCO ActiveSpaces Developer's Guide*.

See Also [tibas_SetMemberInvocable\(\)](#), [tibasSpace_Invoke\(\)](#)

tibas_SetMemberInvocable()

Function

Declaration	<code>tibas_status</code> <code>tibas_SetMemberInvocable</code> (const char* name, tibas_memberInvocable invocable)						
Purpose	Specifies the name of an application function to be invoked on a specified space member.						
Parameters	<table><tr><th>Parameter</th><th>Description</th></tr><tr><td>name</td><td>Specifies the name of an application function to be invoked.</td></tr><tr><td>invocable</td><td>Specifies a <code>tibas_memberInvocable</code> structure that specifies the space and context for the invocation, and which also returns the result of the invocation.</td></tr></table>	Parameter	Description	name	Specifies the name of an application function to be invoked.	invocable	Specifies a <code>tibas_memberInvocable</code> structure that specifies the space and context for the invocation, and which also returns the result of the invocation.
Parameter	Description						
name	Specifies the name of an application function to be invoked.						
invocable	Specifies a <code>tibas_memberInvocable</code> structure that specifies the space and context for the invocation, and which also returns the result of the invocation.						
Remarks	<p>Use the <code>tibas_SetMemberInvocable()</code> function to specify the name of an application function to be invoked on a specified space member.</p> <p>You must call <code>tibas_SetMemberInvocable()</code> before you call the tibasSpace_InvokeMember() function.</p> <p>The name parameter specifies the name of the invocable function that is called.</p> <p>The invocable parameter specifies a <code>tibas_memberInvocable</code> type that provides the name of a function that conforms to the function prototype for invocable functions.</p> <p>The <code>tibas_memberInvocable</code> typedef specifies the function prototype for an invocable function, and has the following syntax:</p> <pre>typedef void (TIBAS_CALL_API *tibas_memberInvocable)(tibasSpace space, tibasTuple context, tibasTuple result);</pre> <p>where:</p> <ul style="list-style-type: none">space Specifies the space on which the invocable function is to be invoked.context Specifies a user-defined context string used to identify the function.result Returns the result of the invocation.						

For a sample program showing the use of `tibas_SetMemberInvocable()` and `tibas_SpaceInvokeMember()`, see the **InvokeClient** sample program provided with the ActiveSpaces distribution. For documentation on `InvokeClient`, see [Remote Space Invocation: InvokeClient](#) in the *TIBCO ActiveSpaces Developer's Guide*.

See Also [tibas_SetInvocable\(\)](#), [tibasSpace_InvokeMember\(\)](#)

tibas_SetLogLevel()

Function

Declaration `tibas_status` `tibas_SetLogLevel(`
 `tibas_logLevel` `logLevel)`

Purpose Sets the log level used by ActiveSpaces.

Parameters	Parameter	Description
	<code>logLevel</code>	The log level to set. The possible values are <code>TIBAS_LOG_FATAL</code> , <code>TIBAS_LOG_ERROR</code> , <code>TIBAS_LOG_WARN</code> , <code>TIBAS_LOG_FINE</code> , <code>TIBAS_LOG_FINER</code> , <code>TIBAS_LOG_FINEST</code> , or <code>TIBAS_LOG_INFO</code> .

Remarks Use the `tibas_SetLogLevel()` function to set the log level used by ActiveSpaces.

See Also [tibas_GetLogLevel\(\)](#), [tibas_EnableFileLogging\(\)](#), [tibas_DisableFileLogging\(\)](#)

tibas_GetLogLevel()

Function

Declaration `tibas_status` `tibas_GetLogLevel`(
 `tibas_logLevel*` `logLevel`)

Purpose Returns the log level used by ActiveSpaces.

Parameters	Parameter	Description
	<code>logLevel</code>	The log level returned by the function. The possible values are <code>TIBAS_LOG_FATAL</code> , <code>TIBAS_LOG_ERROR</code> , <code>TIBAS_LOG_WARN</code> , <code>TIBAS_LOG_FINE</code> , <code>TIBAS_LOG_FINER</code> , <code>TIBAS_LOG_FINEST</code> , or <code>TIBAS_LOG_INFO</code> .

Remarks Use the `tibas_GetLogLevel()` function to return the log level that is used by ActiveSpaces.

See Also [tibas_SetLogLevel\(\)](#), [tibas_EnableFileLogging\(\)](#), [tibas_DisableFileLogging\(\)](#)

tibas_EnableFileLogging()

Function

Declaration `tibas_status` `tibas_EnableFileLogging`(
 `const char*` `logDir`,
 `const char*` `fileName`,
 `tibas_logLevel` `logLevel`)

Purpose Enables logging of specified log level messages to a specified log file.

Parameters	Parameter	Description
	<code>logDir</code>	Specify the directory path where you want the log file written.
	<code>fileName</code>	Specify the name of the log file.
	<code>logLevel</code>	Specify the log level of the messages to write to the specified log file. The possible values are <code>TIBAS_LOG_FATAL</code> , <code>TIBAS_LOG_ERROR</code> , <code>TIBAS_LOG_WARN</code> , <code>TIBAS_LOG_FINE</code> , <code>TIBAS_LOG_FINER</code> , <code>TIBAS_LOG_FINEST</code> , or <code>TIBAS_LOG_INFO</code> .

Remarks Use the `tibas_EnableFileLogging()` function to enable logging of TIBCO ActiveSpaces messages to a log file.

 You can specify the log file directory, the name of the log file, and the level of logging messages to add to the file.

 You can call this function multiple times to specify writing of different log level messages to separate log files. For example, you can specify that `TIBAS_LOG_ERROR` messages are written to one log file and `TIBAS_LOG_INFO` messages are written to another log file.

 After you have enable file logging, you can disable it if necessary by calling the `tibas_DisableFileLogging()` function.

See Also `tibas_DisableFileLogging()`

tibas_DisableFileLogging()

Function

Declaration	<code>tibas_status</code> <code>tibas_DisableFileLogging()</code>
Purpose	Disables file logging.
Parameters	None
Remarks	Use the <code>tibas_DisableFileLogging()</code> function to disable file logging.
See Also	tibas_EnableFileLogging()

Chapter 26 **StringList**

This chapter explains the StringList operations.

Topics

- [StringList Operations, page 486](#)

StringList Operations

The following table lists the StringList operations:

Table 32 StringList

Function or Type	Description	Page
tibasStringList_Free()	Frees a specified StringList object.	487
tibasStringList_Get()	Returns the string at a specified index position in a string list.	488
tibasStringList_Size()	Returns the number of strings in the a specified string list.	489

tibasStringList_Free()

Function

Declaration `tibas_status` tibasStringList_Free(tibasStringList* stringList)

Purpose Frees a specified StringList object.

Parameters	Parameter	Description
	stringList	The StringList object on which the operation is invoked.

Remarks Use the tibasStringList_Free() function to free a specified stringList object.

See Also `tibasStringList_Get()`, `tibasStringList_Size()`

tibasStringList_Get()

Function

Declaration `tibas_status tibasStringList_Get(
 tibasStringList stringList,
 const char** string,
 tibas_int index)`

Purpose Returns the string at a specified index position in a string list.

Parameters	Parameter	Description
	stringList	The StringList object on which the operation is invoked.
	string	The string returned by the operation.
	index	The index position of the string to be returned by the operation.

Remarks Use the `tibasStringList_Get()` function to return the string at a specified index position in a specified `stringList` object.

See Also [tibasStringList_Size\(\)](#), [tibasStringList_Free\(\)](#)

tibasStringList_Size()

Function

```
Declaration      tibas_status tibasStringList_Size
                   (tibasStringList stringList,
                    tibas_int* size)
```

Purpose Returns the number of strings in the a specified string list.

Parameters

Parameter	Description
stringList	The StringList object on which the operation is invoked.
size	The number of strings in the list.

Remarks	Use the <code>tibasStringList_Size()</code> function to determine the size of a specified string list.
----------------	--

See Also [tibasStringList_Get\(\)](#), [tibasStringList_Free\(\)](#)

Chapter 27 **Error**

This chapter explains the Error operations.

Topics

- [Error Operations](#)

Error Operations

The following table lists the Error operations:

Table 33 Error

Function or Type	Description	Page
tibasError_GetError()	Returns the error stack if there was an error or a severe error, as indicated by the status code returned by the operation.	493
tibasError_GetCode()	Returns the error code for a specified error.	495
tibasError_GetMessage()	Returns the error message related to a specified error.	496
tibasError_GetStackTrace()	Returns a stack trace for a specified error.	497
tibasError_Free()	Destroys allocated resources and frees memory used by a specified error.	498

tibasError_GetError()

Function

Declaration	<code>tibas_status</code> <code>tibasError_GetError</code> (<code>tibasError*</code> <code>error</code>)					
Purpose	Returns the error stack if there was an error or a severe error, as indicated by the status code returned by the operation.					
Parameters	<table><tr><th>Parameter</th><th>Description</th></tr><tr><td><code>error</code></td><td>The error stack returned by the function that last experienced an error or a severe error.</td></tr></table>		Parameter	Description	<code>error</code>	The error stack returned by the function that last experienced an error or a severe error.
Parameter	Description					
<code>error</code>	The error stack returned by the function that last experienced an error or a severe error.					
Remarks	Use the <code>tibasError_GetError()</code> function to return the error stack for a specified error.					
See Also	<code>tibasError_GetSevereError()</code> ,					

tibasError_GetSevereError()

Function

Declaration	<code>tibas_status</code> <code>tibasError_GetSevereError</code> (<code>tibasError* error</code>)					
Purpose	Returns the error stack if there was a severe error, as indicated by the status code returned by the operation.					
Parameters	<table><tr><th>Parameter</th><th>Description</th></tr><tr><td><code>error</code></td><td>The error stack returned by the function that last experienced a severe error.</td></tr></table>		Parameter	Description	<code>error</code>	The error stack returned by the function that last experienced a severe error.
Parameter	Description					
<code>error</code>	The error stack returned by the function that last experienced a severe error.					
Remarks	Use the <code>tibasError_GetSevereError()</code> function to return the error stack for a function that caused a severe error.					
See Also	<code>tibasError_GetError()</code>					

tibasError_GetCode()

Function

Declaration `tibas_status` `tibasError_GetCode`
 (`tibasError` `error`,
 `tibas_status*` `code`)

Purpose Returns the error code for a specified error.

Parameters	Parameter	Description
	<code>error</code>	The error stack.
	<code>code</code>	The error code returned by the function.

Remarks Use the `tibasError_GetCode()` function to return the error code for a specified error.

See Also [tibasError_GetMessage\(\)](#)

tibasError_GetMessage()

Function

Declaration `tibas_status` `tibasError_GetMessage`
 (`tibasError` `error`,
 `char**` `message`)

Purpose Returns the error message related to a specified error.

Parameters

Parameter	Description
<code>error</code>	The error stack.
<code>message</code>	The error code returned by the function. The message data is owned by the Error object. There is no need to free it explicitly, as it will be freed when the Error object is freed.

Remarks Use the `tibasError_GetMessage()` function to return the error message associated with a specified error.

See Also [tibasError_GetCode\(\)](#)

tibasError_GetStackTrace()

Function

Declaration `tibas_status` `tibasError_GetStackTrace`
 (`tibasError` `error`,
 `char**` `stackTrace`)

Purpose Returns a stack trace for a specified error.

Parameters	Parameter	Description
	<code>error</code>	The stack of errors to iterate through.
	<code>stackTrace</code>	A string representation of the stack trace.

Remarks Use the `tibasError_GetStackTrace()` function to return the stack trace for a specified error.

 The stack trace is owned by the error object. There is no need to free it explicitly, as it will be freed when the error object is freed.

See Also [tibasError_GetCode\(\)](#), [tibasError_GetMessage\(\)](#)

tibasError_Free()

Function

Declaration	<code>tibas_status</code> <code>tibasError_Free</code> (<code>tibasError</code> <code>error</code>)	
Purpose	Destroys allocated resources and frees memory used by a specified error.	
Parameters	Parameter	Description
	<code>error</code>	The stack of errors to destroy.
Remarks	Use the <code>tibasError_Free()</code> function to free the memory and resources used by a specified error.	
See Also	tibasError_GetError()	

Appendix A **Error Codes**

This appendix lists error codes used in TIBCO ActiveSpaces.

Topics

- [Error Codes](#)

Error Codes

The following table lists the error codes that can be contained in the error objects.

Table 1 Error Codes

Constant	Error Code	Description
ADMIN_INVALID_CMD_EXCEPTION	10001	Admin command syntax error
ADMIN_INVALID_EXCEPTION	10002	Invalid admin handle
ADMIN_METASPACE_CONNECT_EXCEPTION	10003	Please connect to metaspace first
BROWSER_CREATE_EXCEPTION	20001	Browser creation error
BROWSER_INVALID_EXCEPTION	20002	Invalid browser handle
BROWSER_LOCK_EXCEPTION	20003	Lock browser next() error
BROWSER_NEXT_EXCEPTION	20004	Get browser next() error
BROWSER_TAKE_EXCEPTION	20005	Take browser next() error
BROWSER_INVALID_TYPE_EXCEPTION	20006	Invalid browser type
BROWSERDEF_INVALID_EXCEPTION	30001	Invalid browser def
COMMON_INIT_EXCEPTION	40001	Admin command syntax error
COMMON_INVALID_EXCEPTION	40002	Invalid admin handle
ENTRY_INVALID_EXCEPTION	50001	Invalid entry handle
ENTRY_INVALID_LOCK_EXCEPTION	50002	Invalid entry lock handle
ENTRY_SERIALIZE_ERROR	50003	Entry serialization error
ENTRY_LOCKED_EXCEPTION	50004	Entry is locked
ENTRY_NOT_LOCKED_EXCEPTION	50005	Entry can not be unlocked as it's lock has already expired or it has already been unlocked
ERROR_INVALID_EXCEPTION	60001	Invalid error handle

Table 1 Error Codes

Constant	Error Code	Description
EVENT_INVALID_EXCEPTION	70001	Invalid event handle
FACTORY_INVALID_EXCEPTION	80001	Invalid factory handle
FACTORY_CREATE_FAILURE_EXCEPTION	80002	Unable to create the factory
FILTER_INVALID_EXCEPTION	90001	Invalid filter handle
ITERATOR_EVENT_EXCEPTION	11001	Event browser next() error
ITERATOR_INVALID_EXCEPTION	11002	Invalid iterator handle
LISTENERDEF_INVALID_EXCEPTION	12001	Invalid listener def
MEMBER_INVALID_EXCEPTION	13001	Invalid member handle
MEMBER_ITERATOR_INVALID_EXCEPTION	13002	Member iterator invalid handle
METASPACE_CONNECT_EXCEPTION	14001	metaspace connection error
METASPACE_INVALID_EXCEPTION	14002	Invalid metaspace handle
METASPACE_INVALID_NAME_EXCEPTION	14003	Invalid metaspace name
METASPACE_LEAVE_EXCEPTION	14004	Internal metaspace leave error
METASPACE_URL_EXCEPTION	14005	Internal metaspace URL error
METASPACE_SPACE_NAMES_EXCEPTION	14006	Metaspace get space names error
METASPACE_NOT_CONNECTED_EXCEPTION	14007	Metaspace is not connected
METASPACE_ALREADY_CONNECTED_EXCEPTION	14008	Already connected to this metaspace
NETWORK_TIMEOUT_EXCEPTION	15001	Network operation timeout
NETWORK_TCP_NO_PORT_AVAILABLE_EXCEPTION	15002	No TCP port available to create a socket on
NETWORK_TCP_BAD_INTERFACE_EXCEPTION	15003	Unable to create a socket on the interface specified in the URL
NETWORK_MULTICAST_RV_ERROR_EXCEPTION	15004	Transport error

Table 1 Error Codes

Constant	Error Code	Description
NETWORK_MULTICAST_PGM_DATALOSS_EXCEPTION	15005	TIBPGM potential dataloss error
SPACE_BUFFER_EXCEPTION	16001	Buffer output error
SPACE_CREATE_EXCEPTION	16002	Internal space create error
SPACE_EXISTS_EXCEPTION	16003	Space already exists.
SPACE_GET_EXCEPTION	16004	Internal space get error
SPACE_INVALID_EXCEPTION	16005	Invalid space handle
SPACE_INVALID_ENTRY_EXCEPTION	16006	Invalid entry handle
SPACE_INVALID_NAME_EXCEPTION	16007	Invalid space name
SPACE_JOIN_EXCEPTION	16008	Internal space join error
SPACE_LEAVE_EXCEPTION	16009	Internal space leave error
SPACE_LOCK_EXCEPTION	16010	Internal space lock error
SPACE_NONE_EXIST_EXCEPTION	16011	No user spaces exist
SPACE_OPERATION_NOT_SUPPORTED_EXCEPTION	16012	Operation not supported
SPACE_PUT_EXCEPTION	16013	Internal space put error
SPACE_REFRESH_EXCEPTION	16014	Internal space refresh error
SPACE_REMOVE_EXCEPTION	16015	Internal space remove error
SPACE_SIZE_EXCEPTION	16016	Internal space size error
SPACE_TAKE_EXCEPTION	16017	Internal space take error
SPACE_UNLOCK_EXCEPTION	16018	Internal space unlock error
SPACE_UPDATE_EXCEPTION	16019	Internal space update error
SPACE_NO_DATA_EXCEPTION	16020	No matching tuple found in the Space

Table 1 Error Codes

Constant	Error Code	Description
SPACE_UPDATE_FAILURE_EXCEPTION	16021	Update failing because existing tuple in the space does not match the old tuple
SPACE_UPDATE_KEY_MISMATCH_EXCEPTION	16022	Update failing because the new tuple's key field(s) must match old tuple's key field(s)
SPACE_UNDEFINED_EXCEPTION	16023	The space has not been defined in the metaspace (or has been dropped)
SPACE_NOT_ENOUGH_SEEDERS_EXCEPTION	16024	There are not enough Seeders currently joined in the space to allow the operation
SPACE_DEFINITION_MISMATCH_EXCEPTION	16025	The space has already been defined in the metaspace with attributes that do not match those specified in this request
SPACE_DROP_FAILURE_EXCEPTION	16026	Unable to drop the space because it still has Members using it
SPACEDEF_DESERIALIZE_EXCEPTION	17001	Internal space def error
SPACEDEF_GET_EXCEPTION	17002	Internal space def get error
SPACEDEF_INVALID_EXCEPTION	17003	Invalid space def handle
SPACEDEF_INVALID_POLICY_EXCEPTION	17004	Invalid space def policy
SPACEDEF_INVALID_ROLE_EXCEPTION	17005	Invalid space def role
SPACEDEF_UNMARSHAL_EXCEPTION	17006	Internal space def error
SPACELISTENER_START_EXCEPTION	18001	Space listener start error
SPACELISTENER_INVALID_EXCEPTION	18002	Invalid space listener handle
SPACEMGR_GET_EXCEPTION	19001	Internal space manager error
SPACEMGR_INVALID_EXCEPTION	19002	Invalid space manager handle

Table 1 Error Codes

Constant	Error Code	Description
SPACEMGR_SPACENAMES_EXCEPTION	19003	Internal get space names error
TUPLEDEF_INVALID_EXCEPTION	21001	Invalid tuple def handle
TUPLEDEF_DESERIALIZE_EXCEPTION	21002	Internal tuple def error
TUPLEDEF_UNMARSHAL_EXCEPTION	21003	Key field is not defined
TUPLEDEF_NO_KEY_EXCEPTION	21004	Internal tuple def error
TUPLE_BAD_FIELD_NAME_EXCEPTION	22001	Field name not allowed as it is not defined
TUPLE_BAD_FIELD_TYPE_EXCEPTION	22002	Field type is not a valid type
TUPLE_BAD_KEY_TYPE_EXCEPTION	22003	Key field type is not a valid type
TUPLE_FIELD_TYPE_MISMATCH_EXCEPTION	22004	Tuple field is missing
TUPLE_INVALID_EXCEPTION	22005	Type of one of the tuple's fields is incompatible with the space's definition
TUPLE_INVALID_FIELD_EXCEPTION	22006	Invalid tuple handle
TUPLE_KEY_FIELD_EXCEPTION	22007	Field type mismatch
TUPLE_FIELD_MISSING_EXCEPTION	22008	Tuple key field missing
TUPLE_MULTIBYTE_CHAR_EXCEPTION	22009	Multibyte char not supported
TUPLE_NOT_ENOUGH_FIELDS_EXCEPTION	22010	Tuple does not have enough fields
TUPLE_SERIALIZE_EXCEPTION	22011	Tuple decoding error
INTERNAL_ERROR_EXCEPTION	99999	Internal error

Appendix B **Datatypes**

This appendix lists C Datatypes and enumerated types used in TIBCO ActiveSpaces.

Topics

- [C Datatypes](#)
- [Enumerated Types](#)

C Datatypes

Table 2 C Datatypes

Type	Description
tibas_short	short
tibas_int	int
tibas_long	long
tibas_float	float
tibas_double	double
_tibasDateTime	<p>A structure that defines the time format used in TIBCO ActiveSpaces:</p> <p>The _tibasDateTime struct is defined as follows:</p> <pre>struct _tibasDateTime { tibas_long sec; tibas_int nsec; };</pre> <p>The members in the _tibas_dateTime struct are defined as follows:</p> <ul style="list-style-type: none">• sec Specifies the time, in seconds, since the UNIX EPOCH time (1970-01-01 00:00:00 +0000).• nsec Specifies the time, in nanoseconds, since the UNIX EPOCH time (1970-01-01 00:00:00 +0000). This field provides is extra timing information in addition to the number of seconds since the EPOCH time; usually this is set to 0.

Enumerated Types

The following table lists the enumerated types used by TIBCO ActiveSpaces functions.:

Table 3 Enumerated Types

Function or Type	Description	Page
tibas_boolean	This type represents the boolean data returned by TIBCO ActiveSpaces functions.	509
tibas_loglevel	This type represents the log levels used by TIBCO ActiveSpaces functions.	510
tibas_status	This type represents the return codes used by TIBCO ActiveSpaces functions.	511
tibas_type	This type represents the datatypes used by TIBCO ActiveSpaces functions.	512
tibas_indexType	This type represents the datatypes used to determine the type of indexing used by ActiveSpaces.	513
tibas_distributionPolicy	This type represents the datatypes used by TIBCO ActiveSpaces distribution policy.	515
tibas_evictionPolicy	This type represents the eviction policy used by TIBCO ActiveSpaces spaces.	515
tibas_distributionRole	This type represents the roles that can be assigned to the TIBCO ActiveSpaces space member.	516
tibas_persistenceType	This type represents the datatypes used by to determine the type of persistence used by ActiveSpaces.	517
tibas_persistencePolicy	This type specifies the type of data transmission used to implement persistence.	518
tibas_managementRole	This type specifies the management role that can be assigned to a member of the metaspace.	519
tibas_eventType	This type represents the event types used by TIBCO ActiveSpaces functions.	520

Table 3 Enumerated Types

Function or Type	Description	Page
tibas_memberEventType	This type represents the type of event returned by a member action.	521
tibas_spaceState	This type indicates the state of a space.	522
tibas_lockType	This type specifies the type of lock used with space entries.	523
tibas_lockScope	This type specifies the lock scope value that is set for a specified space.	524
tibas_browserType	This type specifies the type of browser that can be set up using the TIBCO ActiveSpaces functions.	525
tibas_distributionScope	This type represents the datatypes used by TIBCO ActiveSpaces functions that work with the distribution scope of a space.	526
tibas_updateTransport	This type represents the data protocol used by TIBCO ActiveSpaces functions.	527
tibas_timeScope	This type represents the datatypes used by TIBCO ActiveSpaces functions that work with the time scope of a space.	528
tibas_actionType	This type represents the types of action performed on Spaces.	529
tibas_opType	This type represents the types of operation performed on Spaces.	530
tibas_authenticationMethod	Indicates the type of authentication method used when TIBCO ActiveSpaces security is implemented and an authentication value is set in the security policy file for a domain.	531

tibas_boolean

Type

- Declaration

```
typedef enum {  
    TIBAS_FALSE,  
    TIBAS_TRUE  
} tibas_boolean;
```
- Purpose

This type represents the boolean data returned by TIBCO ActiveSpaces functions.

Values	Value	Description
	TIBAS_FALSE	The function returned TRUE.
	TIBAS_TRUE	The function returned FALSE.

tibas_loglevel

Type

Declaration

```
typedef enum _tibas_logLevel {
    TIBAS_LOG_FATAL = 0,
    TIBAS_LOG_ERROR = 1,
    TIBAS_LOG_WARN = 2,
    TIBAS_LOG_INFO = 3,
    TIBAS_LOG_FINE = 4,
    TIBAS_LOG_FINER? = 5,
    TIBAS_LOG_FINEST? = 6
} tibas_logLevel;
```

Purpose

This type represents the log levels used by TIBCO ActiveSpaces functions.

Values

Value	Description
TIBAS_LOG_FATAL	Fatal errors output to stdout.
TIBAS_LOG_ERROR	Error messages output to the log.
TIBAS_LOG_WARN	Debug information output to stdout.
TIBAS_LOG_INFO	Warnings output to stdout.
TIBAS_LOG_FINE	Log messages have fine granularity.
TIBAS_LOG_FINER	Log messages are more detailed than with TIBAS_LOG_FINE.
TIBAS_LOG_FINEST	Log messages contain the most detailed information.

tibas_status

Type

Declaration	<pre>typedef enum _tibas_status { TIBAS_OK = 0, TIBAS_NOT_FOUND = 1, TIBAS_ALREADY_EXISTS = 2, TIBAS_LOCKED = 3, TIBAS_MISMATCHED_LOCK = 4, TIBAS_INCOMPATIBLE_TUPLE = 5, TIBAS_MISMATCHED_TUPLE = 6, TIBAS_INCOMPATIBLE_TYPE = 7, TIBAS_LIMIT_EXCEEDED = 8, TIBAS_REMOTE_ERROR = 100, TIBAS_SYS_ERROR = 256, TIBAS_INVALID_ARG = 257, } tibas_status;</pre>
Purpose	This type represents the return codes used by TIBCO ActiveSpaces functions.
Values	For descriptions of the tibas_status values, see , Status Codes, on page 10.

tibas_type

Type

Declaration

```
typedef enum {
    TIBAS_NULL,
    TIBAS_BOOLEAN ,
    TIBAS_CHAR,
    TIBAS_SHORT,
    TIBAS_INTEGER,
    TIBAS_LONG,
    TIBAS_FLOAT,
    TIBAS_DOUBLE,
    TIBAS_BLOB,
    TIBAS_DATETIME,
    TIBAS_STRING
} tibas_type;
```

Purpose

This type represents the datatypes used by TIBCO ActiveSpaces functions.

Values

Value	Description
TIBAS_NULL	NULL pointer.
TIBAS_BOOLEAN	Returns TIBAS_TRUE or TIBAS_FALSE.
TIBAS_CHAR	Char value.
TIBAS_SHORT	Short value.
TIBAS_INTEGER	Integer value.
TIBAS_LONG	Long integer value.
TIBAS_FLOAT	Float value.
TIBAS_DOUBLE	Double value.
TIBAS_BLOB	A binary large object (BLOB) value.
TIBAS_DATETIME	Date and time value.
TIBAS_STRING	String value.

tibas_indexType

Type

Declaration

```
typedef enum _tibas_indexType {
    TIBAS_INDEX_HASH,
    TIBAS_INDEX_TREE
} tibas_indexType;
```

This type represents the datatypes used to determine the type of indexing used by ActiveSpaces.

Values	Value	Description
	TIBAS_INDEX_HASH	Specifies that indexing is done by using a hash table.
	TIBAS_INDEX_TREE	Specifies that indexing is done using a tree.

tibas_distributionPolicy

Type

Declaration	<pre>typedef enum { TIBAS_NON_DISTRIBUTED , TIBAS_DISTRIBUTED } tibas_distributionPolicy;</pre>							
Purpose	This type represents the datatypes used by TIBCO ActiveSpaces distribution policy.							
Values	<table><tr><th>Value</th><th>Description</th></tr><tr><td>TIBAS_NON_DISTRIBUTED</td><td>Storage of all of the tuples is assigned to a single seeder Member of the space.</td></tr><tr><td>TIBAS_DISTRIBUTED</td><td>Storage of the tuples is distributed amongst seeder Members of the space.</td></tr></table>		Value	Description	TIBAS_NON_DISTRIBUTED	Storage of all of the tuples is assigned to a single seeder Member of the space.	TIBAS_DISTRIBUTED	Storage of the tuples is distributed amongst seeder Members of the space.
Value	Description							
TIBAS_NON_DISTRIBUTED	Storage of all of the tuples is assigned to a single seeder Member of the space.							
TIBAS_DISTRIBUTED	Storage of the tuples is distributed amongst seeder Members of the space.							

tibas_evictionPolicy

Type

Declaration `typedef enum _tibas_evictionPolicy {
 TIBAS_EVICTION_NONE = 0,
 TIBAS_EVICTION_LRU = 1
} tibas_evictionPolicy;`

Purpose This type represents the eviction policy used by TIBCO ActiveSpaces spaces.

Values

Value	Description
TIBAS_EVICTION_NONE	Specifies that no elements are evicted from the space.
TIBAS_EVICTION_LRU	Specifies that tuples are evicted from the space using the “least recently used” algorithm when the capacity of the space is reached

tibas_distributionRole

Type

Declaration

```
typedef enum _tibas_distributionRole {  
    TIBAS_DISTRIBUTION_ROLE_NONE      = 0,  
    TIBAS_DISTRIBUTION_ROLE_LEECH     = 1,  
    TIBAS_DISTRIBUTION_ROLE_SEEDER    = 3  
} tibas_distributionRole;
```

Purpose

This type represents the roles that can be assigned to the TIBCO ActiveSpaces space member.

Values

Value	Description
TIBAS_DISTRIBUTION_ROLE_NONE	Sets persistence to none.
TIBAS_DISTRIBUTION_ROLE_LEECH	The Member is not willing to store any tuples put into the space by any Member.
TIBAS_DISTRIBUTION_ROLE_SEEDER	The Member is willing to store (seed) tuples put into the space by any Member.

tibas_persistenceType

Type

Declaration

```
typedef enum _tibas_persistenceType {  
    TIBAS_PERSISTENCE_NONE           = 0,  
    TIBAS_PERSISTENCE_SHARED_ALL     = 1,  
    TIBAS_PERSISTENCE_SHARED_NOTHING = 2  
} tibas_persistenceType;
```

This type represents the datatypes used by to determine the type of persistence used by ActiveSpaces.

Values

Value	Description
TIBAS_PERSISTENCE_NONE	Specifies that data is not persisted.
TIBAS_PERSISTENCE_SHARED_ALL	Specifies that shared all persistence is used.
TIBAS_PERSISTENCE_SHARED_NOTHING	Specifies that shared-nothing persistence is used.

tibas_persistencePolicy

Type

Declaration

```
typedef enum _tibas_persistencePolicy {  
    TIBAS_PERSISTENCE_SYNC          = 0,  
    TIBAS_PERSISTENCE_ASYNC        = 1  
} tibas_persistencePolicy;
```

Purpose

This type specifies the type of data transmission used to implement persistence.

Values

Value	Description
TIBAS_PERSISTENCE_SYNC	Specifies that persistence is implemented using synchronous methods.
TIBAS_PERSISTENCE_ASYNC	Specifies that persistence is implemented using asynchronous methods.

tibas_managementRole

Type

Declaration

```
typedef enum _tibas_managementRole {  
    TIBAS_MANAGEMENT_ROLE_NONE           = 0,  
    TIBAS_MANAGEMENT_ROLE_MEMBER        = 1,  
    TIBAS_MANAGEMENT_ROLE_MANAGER       = 2  
} tibas_managementRole;
```

Purpose This type specifies the management role that can be assigned to a member of the metaspace.

Values

Value	Description
TIBAS_MANAGEMENT_ROLE_NONE	The member has no management role.
TIBAS_MANAGEMENT_ROLE_MEMBER	The member is simply a member of the Metaspace.
TIBAS_MANAGEMENT_ROLE_MANAGER	The member is currently managing membership for the Metaspace.

tibas_eventType

Type

Declaration

```
typedef enum {
    TIBAS_EVENT_PUT ,
    TIBAS_EVENT_TAKE,
    TIBAS_EVENT_EXPIRE,
    TIBAS_EVENT_EVICT
    TIBAS_EVENT_SEED,
    TIBAS_EVENT_UNSEED
} tibas_eventType;
```

Purpose

This type represents the event types used by TIBCO ActiveSpaces functions.

Values

Value	Description
TIBAS_EVENT_PUT	The Put event type is returned when a member <i>puts</i> or <i>updates</i> a tuple into a space.
TIBAS_EVENT_TAKE	The Take event type is returned when a member <i>takes</i> a tuple from a space.
TIBAS_EVENT_EXPIRE	The Expire event type is returned when a tuple <i>expires</i> from the space.
TIBAS_EVENT_EVICT	The Evict event type is returned when a tuple is evicted from the space.
TIBAS_EVENT_SEED	The Seed event type is returned when a tuple is <i>seeded</i> by the space.
TIBAS_EVENT_UNSEED	The Unseed event type is returned when a tuple into a space is <i>unseeded</i> from the space.

tibas_memberEventType

Type

Declaration

```
typedef enum _tibas_memberEventType {  
    TIBAS_MEMBER_EVENT_JOIN          = 0,  
    TIBAS_MEMBER_EVENT_UPDATE        = 1,  
    TIBAS_MEMBER_EVENT_LEAVE         = 2  
} tibas_memberEventType;
```

Purpose

This type represents the type of event returned by a member action.

Values

Value	Description
TIBAS_MEMBER_EVENT_JOIN	This event type is returned when a member a joins a space.
TIBAS_MEMBER_EVENT_UPDATE	This event type is returned when a member a updates a space.
TIBAS_MEMBER_EVENT_LEAVE	This event type is returned when a member a leaves a space.

tibas_spaceState

Type

Declaration

```
typedef enum _tibas_spaceState {
    TIBAS_SPACE_STATE_INITIAL          = 0,
    TIBAS_SPACE_STATE_LOADING          = 1,
    TIBAS_SPACE_STATE_READY            = 2,
    TIBAS_SPACE_STATE_SUSPENDED        = 3,
    TIBAS_SPACE_STATE_RECOVER          = 4,
    TIBAS_SPACE_STATE_INVALID          = 100
} tibas_spaceState;
```

Purpose This type indicates the state of a space.

Values	Value	Description
	TIBAS_SPACE_STATE_INITIAL	Indicates that a space is initializing.
	TIBAS_SPACE_STATE_LOADING	Indicates that a persisted space is loading.
	TIBAS_SPACE_STATE_READY	Indicates that a space is ready and can be seeded or have space operations performed on it.
	TIBAS_SPACE_STATE_SUSPENDED	Indicates that a space is suspended.
	TIBAS_SPACE_STATE_RECOVER	Indicates that a space is recovering after being in the SUSPENDED state.
	TIBAS_SPACE_STATE_INVALID	Indicates that a space is in an invalid state.

tibas_lockType

Type

Declaration

```
typedef enum _tibas_lockType {  
    TIBAS_LOCK_NONE                = 0,  
    TIBAS_LOCK_EXCLUSIVE            = 1  
} tibas_lockType;
```

Purpose

This type specifies the type of lock used with space entries.

Values

Value	Description
TIBAS_LOCK_NONE	Lock type indicating that the SpaceEntry is not locked.
TIBAS_LOCK_EXCLUSIVE	Lock type indicating that the SpaceEntry is locked (read only) exclusively.

tibas_lockScope

Type

Declaration

```
typedef enum _tibas_lockScope {  
    TIBAS_LOCK_SCOPE_NONE           = 0,  
    TIBAS_LOCK_SCOPE_THREAD         = 1,  
    TIBAS_LOCK_SCOPE_PROCESS        = 2  
} tibas_lockScope;
```

Purpose

This type specifies the lock scope value that is set for a specified space.

Values

Value	Description
TIBAS_LOCK_SCOPE_NONE	Specifies that the lock does not have a scope.
TIBAS_LOCK_SCOPE_THREAD	Specifies that threads cannot write over data that is locked by another thread.
TIBAS_LOCK_SCOPE_PROCESS	Specifies that processes cannot write over data that is locked by another process; however, threads within the same process can overwrite data locked by another thread in the process.

tibas_browserType

Type

Declaration

```

typedef enum _tibas_browserType {
    TIBAS_BROWSER_GET           = 0,
    TIBAS_BROWSER_TAKE         = 1,
    TIBAS_BROWSER_LOCK         = 2
} tibas_browserType;

```

Purpose

This type specifies the type of browser that can be set up using the TIBCO ActiveSpaces functions.

Values

Value	Description
TIBAS_BROWSER_GET	Specifies that the browser will perform GET operations on browsed tuples.
TIBAS_BROWSER_TAKE	Specifies that the browser will perform TAKE operations on browsed tuples.
TIBAS_BROWSER_LOCK	Specifies that the browser will perform LOCK operations on browsed tuples.

tibas_distributionScope

Type

Declaration typedef enum {
 TIBAS_DISTRIBUTION_SCOPE_ALL ,
 TIBAS_DISTRIBUTION_SCOPE_SEEDED
 } tibas_distributionScope;

Purpose This type represents the datatypes used by TIBCO ActiveSpaces functions that work with the distribution scope of a space.

Values	Value	Description
	TIBAS_DISTRIBUTION_SCOPE_ALL	Browse through all the tuples in space.
	TIBAS_DISTRIBUTION_SCOPE_SEEDED	Browse only through the tuples in space that are seeded by this Member (will be empty unless the Member is a seeder on the space).

tibas_updateTransport

Type

Declaration

```
typedef enum tibas_updateTransport {  
    TIBAS_TRANSPORT_UNICAST          = 0,  
    TIBAS_TRANSPORT_MULTICAST        = 1  
} tibas_updateTransport;
```

Purpose

This type represents the data protocol used by TIBCO ActiveSpaces functions.

Values

Value	Description
TIBAS_TRANSPORT_UNICAST	Specifies that unicast is used
TIBAS_TRANSPORT_MULTICAST	Specifies that IP multicast is used.

tibas_timeScope

Type

Declaration `typedef enum {
 TIBAS_TIME_SCOPE_ALL ,
 TIBAS_TIME_SCOPE_SNAPSHOT,
 TIBAS_TIME_SCOPE_NEW,
 TIBAS_TIME_SCOPE_NEW_EVENTS
 } tibas_timeScope;`

Purpose This type represents the datatypes used by TIBCO ActiveSpaces functions that work with the time scope of a space.

Values	Value	Description
	TIBAS_TIME_SCOPE_ALL	The listener starts with all the tuples currently in the space at creation time (which will be presented as an initial set of PUT events) and then is continuously updated according to changes in the space.
	TIBAS_TIME_SCOPE_SNAPSHOT	The listener contains only PUT events corresponding to the tuples stored in the space at creation time.
	TIBAS_TIME_SCOPE_NEW	The listener starts empty and is updated only with events related to new or updated tuples in the space.
	TIBAS_TIME_SCOPE_NEW_EVENTS	The listener starts empty and is updated with all events that occur in the space after creation time.

tibas_actionType

Type

Declaration

```
typedef enum _tibas_actionType {
    TIBAS_ACTION_OPEN           = 0,
    TIBAS_ACTION_LOAD           = 1,
    TIBAS_ACTION_WRITE          = 2,
    TIBAS_ACTION_READ           = 3,
    TIBAS_ACTION_CLOSE          = 4
} tibas_actionType;
```

Purpose

This type represents the types of action performed on Spaces.

Values

Value	Description
TIBAS_ACTION_OPEN	Specifies an Open action
TIBAS_ACTION_LOAD	Specifies a Load action
TIBAS_ACTION_WRITE	Specifies a write action
TIBAS_ACTION_READ	Specifies a read action
TIBAS_ACTION_CLOSE	Specifies a close action
TIBAS_ACTION_ALTER	Specifies an alter space action.

tibas_opType

Type

Declaration

```
typedef enum _tibas_opType {  
    TIBAS_OP_PUT                = 0,  
    TIBAS_OP_TAKE              = 1  
} tibas_opType;
```

Purpose

This type represents the types of operation performed on Spaces.

Values

Value	Description
TIBAS_OP_PUT	Specifies a Put action
TIBAS_OP_TAKE	Specifies a Take action

tibas_authenticationMethod

Type

Declaration	<pre>typedef enum _tibas_authenticationMethod { AUTH_USERPWD, AUTH_X509V3 } tibas_authenticationMethod;</pre>							
Purpose	Indicates the type of authentication method used when TIBCO ActiveSpaces security is implemented and an authentication value is set in the security policy file for a domain.							
Values	<table><tr><th>Value</th><th>Description</th></tr><tr><td>AUTH_USERPWD</td><td>Indicates that the authentication method is user password authentication.</td></tr><tr><td>AUTH_X509V3</td><td>Indicates that the authentication method is X509 v3 authentication.</td></tr></table>		Value	Description	AUTH_USERPWD	Indicates that the authentication method is user password authentication.	AUTH_X509V3	Indicates that the authentication method is X509 v3 authentication.
Value	Description							
AUTH_USERPWD	Indicates that the authentication method is user password authentication.							
AUTH_X509V3	Indicates that the authentication method is X509 v3 authentication.							

Index

A

Action, Op, OpList, and ActionResult Operations [456](#)
Admin Operations [126](#)

B

Browser Operations [274](#)
BrowserDef Operations [262](#)

C

C Datatypes [506](#)
ConnectionDef Operations [192](#)
customer support [xix](#)

E

Enumerated Types [507](#)
Error Codes [500](#)
Error Operations [492](#)
EventBrowser Operations [290](#)
EventBrowserDef Operations [278](#)

F

FieldDef Operations [222](#)
FieldDefList Operations [234](#)
Filter Operations [428](#)

L

Listener Operations [304](#)
ListenerDef Operations [294](#)

M

Member Operations [434](#)
MemberList Operations [446](#)
Metaspace Operations [10](#)

P

Persister Operations [452](#)

R

Result Operations [392](#)
ResultList Operations [406](#)

S

Space Operations [60](#)
SpaceDef Operations [134](#)
SpaceEvent Operation [314](#)
StringList Operations [486](#)
support, contacting [xix](#)

T

technical support [xix](#)

`tibas_boolean` [509](#)

`tibas_DisableFileLogging()` [484](#)

`tibas_distributionPolicy` [514](#)

`tibas_distributionRole` [516](#)

`tibas_distributionScope` [526](#)

`tibas_EnableFileLogging()` [483](#)

`tibas_eventType` [520](#), [521](#), [522](#)

`tibas_FreeData()` [58](#)

`tibas_GetLogLevel()` [482](#)

`tibas_GetMetaspace()` [16](#)

`tibas_GetMetaspaceNames()` [17](#)

`tibas_lockType` [523](#)

`tibas_loglevel` [510](#)

`tibas_role` [517](#)

`tibas_SetInvocable()` [477](#)

`tibas_SetLogLevel()` [481](#)

`tibas_SetMemberInvocable()` [479](#)

`tibas_status` [511](#)

`tibas_timeScope` [528](#)

`tibas_type` [512](#)

`tibasAction_Free()` [464](#)

`tibasAction_GetOps()` [463](#)

`tibasAction_GetSpace()` [461](#)

`tibasAction_GetSpaceName()` [462](#)

`tibasAction_GetTuple()` [460](#)

`tibasAction_GetType()` [459](#)

`tibasActionResult_GetStatus()` [474](#)

`tibasActionResult_GetTuple()` [476](#)

`tibasActionResult_SetFailed()` [473](#)

`tibasActionResult_SetTuple()` [475](#)

`tibasAdmin_Create()` [127](#)

`tibasAdmin_Execute()` [129](#)

`tibasAdmin_Free()` [131](#)

`tibasBrowser_Free()` [276](#)

`tibasBrowser_Next()` [275](#)

`tibasBrowserDef_Create()` [263](#)

`tibasBrowserDef_CreateEx()` [264](#)

`tibasBrowserDef_Free()` [272](#)

`tibasBrowserDef_GetDistributionScope()` [266](#)

`tibasBrowserDef_GetTimeout()` [267](#)

`tibasBrowserDef_GetTimeScope()` [268](#)

`tibasBrowserDef_SetDistributionScope()` [269](#)

`tibasBrowserDef_SetTimeout()` [271](#)

`tibasBrowserDef_SetTimeScope()` [270](#)

`tibasConnectionDef_Free()` [195](#)

`tibasConnectionDef_GetDiscovery()` [209](#)

`tibasConnectionDef_GetListen()` [208](#)

`tibasConnectionDef_SetDiscovery()` [206](#)

`tibasConnectionDef_SetListen()` [205](#)

`tibasError_Free()` [498](#)

`tibasError_GetCode()` [495](#)

`tibasError_GetError()` [493](#)

`tibasError_GetMessage()` [496](#)

`tibasError_GetSevereError()` [494](#)

`tibasError_GetStackTrace()` [497](#)

`tibasEventBrowser_Free()` [291](#)

`tibasEventBrowser_Next()` [292](#)

`tibasEventBrowserDef_Create()` [279](#)

`tibasEventBrowserDef_CreateEx()` [280](#)

`tibasEventBrowserDef_Free()` [288](#)

`tibasEventBrowserDef_GetDistributionScope()` [282](#)

`tibasEventBrowserDef_GetTimeout()` [283](#)

`tibasEventBrowserDef_GetTimeScope()` [284](#)

`tibasEventBrowserDef_SetDistributionScope()` [285](#)

`tibasEventBrowserDef_SetTimeout()` [246](#), [253](#), [259](#), [287](#)

`tibasEventBrowserDef_SetTimeScope()` [286](#)

`tibasFieldDef_Create()` [223](#)

`tibasFieldDef_CreateEx()` [225](#)

`tibasFieldDef_Free()` [232](#)

`tibasFieldDef_GetName()` [226](#)

`tibasFieldDef_GetType()` [227](#)

`tibasFieldDef_IsEncrypted()` [231](#)

`tibasFieldDef_IsNullable()` [228](#)

`tibasFieldDef_SetEncrypted()` [230](#)

`tibasFieldDef_SetKey()` [228](#)

`tibasFieldDef_SetNullable()` [229](#)

`tibasFieldDefList_Free()` [237](#)

`tibasFieldDefList_Get()` [236](#)

`tibasFieldDefList_Size()` [235](#)

`tibasFilter_Create()` [429](#)

`tibasFilter_Eval()` [430](#)

`tibasFilter_Free()` [431](#)

`tibasListener_Create()` [305](#)

`tibasListenerDef_Create()` [295](#)

`tibasListenerDef_CreateEx()` [296](#)

`tibasListenerDef_Free()` [302](#)

`tibasListenerDef_GetDistributionScope()` [299](#)

tibasListenerDef_GetTimeScope() 298
 tibasListenerDef_SetDistributionScope() 301
 tibasListenerDef_SetTimeScope() 300
 tibasMember_Free() 435
 tibasMember_GetDistributionRole() 438
 tibasMember_GetManagementRole() 437
 tibasMember_GetName() 436
 tibasMemberDef_GetAuthenticationCallback() 213
 tibasMemberDef_GetDataStore() 201
 tibasMemberDef_GetMemberName() 200
 tibasMemberDef_GetRemoteDiscovery() 203
 tibasMemberDef_GetSecurityPolicyFile() 211
 tibasMemberDef_GetSecurityTokenFile() 212
 tibasMemberDef_GetWorkerThreadCount() 199
 tibasMemberDef_SetAuthenticationCallback() 216
 tibasMemberDef_SetContext() 210
 tibasMemberDef_SetDataStore() 197
 tibasMemberDef_SetMemberName() 196
 tibasMemberDef_SetRemoteDiscovery() 202
 tibasMemberDef_SetRemoteListen() 204
 tibasMemberDef_SetSecurityPolicyFile() 214
 tibasMemberDef_SetSecurityTokenFile() 215
 tibasMemberDef_SetTimeout() 219
 tibasMemberDef_SetWorkerThreadCount() 198
 tibasMemberList_Free() 449
 tibasMemberList_Get() 448
 tibasMemberList_Size() 447
 tibasMemberListener_Create() 308
 tibasMetaspace_AcquireContext() 47
 tibasMetaspace_BeginTransaction() 43
 tibasMetaspace_Browse() 39
 tibasMetaspace_BrowseEvents() 41
 tibasMetaspace_CommitTransaction() 44
 tibasMetaspace_Connect() 14
 tibasMetaspace_ConnectEx() 15
 tibasMetaspace_DefineSpace() 23
 tibasMetaspace_DropSpace() 25
 tibasMetaspace_Free() 57
 tibasMetaspace_Get UserSpaceNames() 22
 tibasMetaspace_GetLogLevel() 52
 tibasMetaspace_GetMembers() 49
 tibasMetaspace_GetRemoteListen() 38
 tibasMetaspace_GetRemoteMembers() 48
 tibasMetaspace_GetSpace() 18
 tibasMetaspace_GetSpaceEx() 20
 tibasMetaspace_GetSpacefMembers() 50
 tibasMetaspace_GetSpaceMembers() 50
 tibasMetaspace_GetSystemSpace() 21
 tibasMetaspace_Listen() 30
 tibasMetaspace_ListenMemberEvents() 33
 tibasMetaspace_ListenRemoteMemberEvents() 35
 tibasMetaspace_ListenSpaceDef() 37
 tibasMetaspace_ListenSpaceMemberEvents() 32
 tibasMetaspace_ListenSpaceRemoteMemberEvents()
 34
 tibasMetaspace_ListenSpaceState() 36
 tibasMetaspace_ReleaseContext() 46
 tibasMetaspace_RollbackTransaction() 45
 tibasOp_Free() 469
 tibasOp_GetOldTuple() 467
 tibasOp_GetTuple() 466
 tibasOp_GetType() 465
 tibasOp_HasOldTuple() 468
 tibasOpList_Free() 472
 tibasOpList_Get() 471
 tibasOpList_Size() 470
 tibasPersister_Create() 453
 tibasPersister_Free() 455
 tibasRemoteMemberListener_Create() 310
 tibasResult_Free() 397
 tibasResult_GetEntry() 393
 tibasResult_GetError() 395
 tibasResult_GetStatus() 394
 tibasResultList_Free() 416
 tibasResultList_Get() 412
 tibasResultList_GetError() 414
 tibasResultList_GetStatus() 413
 tibasResultList_HasError() 408
 tibasResultList_Put() 415
 tibasResultList_Size() 411
 tibasSpace_Browse() 70
 tibasSpace_BrowseEvents() 72
 tibasSpace_Free() 102, 118
 tibasSpace_Get() 66
 tibasSpace_GetAll() 69
 tibasSpace_GetMetaspace() 101
 tibasSpace_GetMetaspaceName() 100
 tibasSpace_GetName() 99
 tibasSpace_GetSpaceDef() 98
 tibasSpace_IsReady() 67

[tibasSpace_Listen\(\)](#) 85
[tibasSpace_Load\(\)](#) 116
[tibasSpace_LoadAll\(\)](#) 117, 118
[tibasSpace_Lock\(\)](#) 103, 104
[tibasSpace_LockAll\(\)](#) 105
[tibasSpace_LockAllEntries\(\)](#) 109
[tibasSpace_Put\(\)](#) 86, 87
[tibasSpace_PutAll\(\)](#) 89, 90
[tibasSpace_Remove\(\)](#) 96
[tibasSpace_SetDistributionRole\(\)](#) 109
[tibasSpace_SetPersister\(\)](#) 110
[tibasSpace_Size\(\)](#) 102
[tibasSpace_Take\(\)](#) 92, 93
[tibasSpace_TakeAll\(\)](#) 95
[tibasSpace_Unlock\(\)](#) 106
[tibasSpace_UnlockAll\(\)](#) 108
[tibasSpace_WaitForReady\(\)](#) 68
[tibasSpaceDef_AddIndexDef\(\)](#) 166
[tibasSpaceDef_Create\(\)](#) 138
[tibasSpaceDef_GetCapacity\(\)](#) 144
[tibasSpaceDef_GetDistributionFields\(\)](#) 160
[tibasSpaceDef_GetDistributionPolicy\(\)](#) 171
[tibasSpaceDef_GetEvictionPolicy\(\)](#) 147
[tibasSpaceDef_GetFieldDef\(\)](#) 159
[tibasSpaceDef_GetFieldDefs\(\)](#) 162
[tibasSpaceDef_GetIndex\(\)](#) 151
[tibasSpaceDef_GetIndexDef\(\)](#) 167
[tibasSpaceDef_GetIndexDefs\(\)](#) 168
[tibasSpaceDef_GetKeyDef\(\)](#) 165
[tibasSpaceDef_GetLockScope\(\)](#) 148
[tibasSpaceDef_GetLockTTL\(\)](#) 181
[tibasSpaceDef_GetLockWait\(\)](#) 183
[tibasSpaceDef_GetMinSeederCount\(\)](#) 153
[tibasSpaceDef_GetName\(\)](#) 140
[tibasSpaceDef_GetNumFields\(\)](#) 150
[tibasSpaceDef_GetReplicationCount\(\)](#) 142
[tibasSpaceDef_GetTTL\(\)](#) 179
[tibasSpaceDef_GetUpdateTransport\(\)](#) 155
[tibasSpaceDef_IsPersisted\(\)](#) 186
[tibasSpaceDef_IsSyncReplicated\(\)](#) 169
[tibasSpaceDef_PutFieldDef\(\)](#) 161
[tibasSpaceDef_RemoveIndexDef\(\)](#) 152
[tibasSpaceDef_SetCapacity\(\)](#) 145
[tibasSpaceDef_SetDistributionFields\(\)](#) 158
[tibasSpaceDef_SetDistributionPolicy\(\)](#) 172
[tibasSpaceDef_SetEvictionPolicy\(\)](#) 146
[tibasSpaceDef_SetKey\(\)](#) 157
[tibasSpaceDef_SetKeyDef\(\)](#) 164
[tibasSpaceDef_SetLockScope\(\)](#) 149
[tibasSpaceDef_SetLockTTL\(\)](#) 182
[tibasSpaceDef_SetLockWait\(\)](#) 184
[tibasSpaceDef_SetMinSeederCount\(\)](#) 154
[tibasSpaceDef_SetName\(\)](#) 141
[tibasSpaceDef_SetPersisted\(\)](#) 185
[tibasSpaceDef_SetReplicationCount\(\)](#) 143
[tibasSpaceDef_SetSyncReplicated\(\)](#) 170
[tibasSpaceDef_SetTTL\(\)](#) 180
[tibasSpaceDef_SetUpdateTransport\(\)](#) 156
[tibasSpaceDef_TakeFieldDef\(\)](#) 163
[tibasSpaceDefListener_Create\(\)](#) 312
[tibasSpaceEvent_Free\(\)](#) 319
[tibasSpaceEvent_GetEntry\(\)](#) 316
[tibasSpaceEvent_GetSpace\(\)](#) 318
[tibasSpaceEvent_GetType\(\)](#) 317
[tibasSpaceEvent_IsNew\(\)](#) 320
[tibasSpaceMemberListener_Create\(\)](#) 307
[tibasSpaceRemoteMemberListener_Create\(\)](#) 309
[tibasSpaceStateListener_Create\(\)](#) 311
[tibasStringList_Free\(\)](#) 487
[tibasStringList_Get\(\)](#) 488
[tibasStringList_Size\(\)](#) 489
[tibasTuple_Create\(\)](#) 348
[tibasTuple_Deserialize\(\)](#) 375
[tibasTuple_Eval\(\)](#) 381
[tibasTuple_Exists\(\)](#) 377
[tibasTuple_Free\(\)](#) 382
[tibasTuple_GetBlob\(\)](#) 358
[tibasTuple_GetBoolean\(\)](#) 349
[tibasTuple_GetChar\(\)](#) 350
[tibasTuple_GetDateTime\(\)](#) 357
[tibasTuple_GetDouble\(\)](#) 355
[tibasTuple_GetFieldNames\(\)](#) 378
[tibasTuple_GetFieldType\(\)](#) 379
[tibasTuple_GetFloat\(\)](#) 354
[tibasTuple_GetInt\(\)](#) 352
[tibasTuple_GetLong\(\)](#) 353
[tibasTuple_GetShort\(\)](#) 351
[tibasTuple_GetString\(\)](#) 356
[tibasTuple_IsNull\(\)](#) 376
[tibasTuple_PutAll\(\)](#) 380

- `tibasTuple_PutBlob()` 369
- `tibasTuple_PutBoolean()` 359
- `tibasTuple_PutChar()` 360
- `tibasTuple_PutDateTime()` 367
- `tibasTuple_PutDouble()` 365
- `tibasTuple_PutFloat()` 364
- `tibasTuple_PutInt()` 362
- `tibasTuple_PutLong()` 363
- `tibasTuple_PutShort()` 361
- `tibasTuple_PutString()` 366
- `tibasTuple_Remove()` 370
- `tibasTuple_Serialize()` 374
- `tibasTuple_Size()` 371
- `tibasTuple_ToString()` 373
- `tibasTupleList_Create()` 385
- `tibasTupleList_Free()` 389
- `tibasTupleList_Get()` 387
- `tibasTupleList_Put()` 388
- `tibasTupleList_Size()` 386
- `TIBCO_HOME` xvii
- Tuple Operations 344
- TupleList Operations 384