

TIBCO ActiveSpaces®

Administration

Software Release 2.4

February 2019

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.

ANY SOFTWARE ITEM IDENTIFIED AS THIRD PARTY LIBRARY IS AVAILABLE UNDER SEPARATE SOFTWARE LICENSE TERMS AND IS NOT PART OF A TIBCO PRODUCT. AS SUCH, THESE SOFTWARE ITEMS ARE NOT COVERED BY THE TERMS OF YOUR AGREEMENT WITH TIBCO, INCLUDING ANY TERMS CONCERNING SUPPORT, MAINTENANCE, WARRANTIES, AND INDEMNITIES. DOWNLOAD AND USE OF THESE ITEMS IS SOLELY AT YOUR OWN DISCRETION AND SUBJECT TO THE LICENSE TERMS APPLICABLE TO THEM. BY PROCEEDING TO DOWNLOAD, INSTALL OR USE ANY OF THESE ITEMS, YOU ACKNOWLEDGE THE FOREGOING DISTINCTIONS BETWEEN THESE ITEMS AND TIBCO PRODUCTS.

This document 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.

TIBCO, the TIBCO logo, Two-Second Advantage, TIB, Information Bus, Rendezvous, and TIBCO Rendezvous are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

Enterprise Java Beans (EJB), Java Platform Enterprise Edition (Java EE), Java 2 Platform Enterprise Edition (J2EE), and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle Corporation in the U.S. and 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. Please see the readme.txt 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.

This and other products of TIBCO Software Inc. may be covered by registered patents. Please refer to TIBCO's Virtual Patent Marking document (<https://www.tibco.com/patents>) for details.

Copyright © 2009-2019. TIBCO Software Inc. All Rights Reserved.

Contents

Figures	8
TIBCO Documentation and Support Services	9
Overview of Administration	10
Deployment Modes	11
Clients (Leeches) and Servers (Seeders)	11
Shared-All Persistence	12
Shared-Nothing Persistence	12
Host-Aware Replication	13
Remote Client Architecture	15
Cross-Site Replication	16
Best Practices for Deploying in a Virtual Environment	17
Best Practices for Node Discovery	18
Admin CLI for ActiveSpaces Administration	20
Launching the Admin CLI	20
Using a Script File to Pass Arguments	21
alter space	22
clear	24
clear set password	24
close	25
connect	25
define create safe_password	28
define create security_policy	29
define create security_token	31
define create space	32
disconnect	38
drop space	38
execute on member	39
execute on members	39
add proxy	40
execute on proxy (deprecated)	40
drop proxy	41
export metaspace	41
help	42
join space	42
leave space	43
open	43

quit exit bye	44
recover metaspace	44
recover space	46
resume space	48
cleanup datastore	48
suspend persistence	49
resume persistence	49
commit transaction	50
rollback transaction	50
set event_log	51
set file_log	52
show describe console_log	53
show describe member	53
show describe members	55
show describe metaspaces	55
show describe space	56
show describe space entries	58
show describe space locks	59
show describe space size	60
show describe space query	60
show describe query	61
show describe spaces	61
show describe transactions	62
show describe event_log	62
show describe file_log	63
show describe member stats	63
show describe system stats	64
abort query	66
unlock	66
validate policy_file	67
validate token_file	68
validate truststore	68
as-admin Commands for Cross-site Replication	69
alter site	69
create site	69
drop site	70
resume site	70
show local site	71
show remote site	71

show sites	71
suspend site	71
site prompt	72
The as-dump Utility	73
The as-agent Utility	75
Starting as-agent	75
as-agent Command Parameters	76
The as-convert Utility	81
Administering ActiveSpaces Security	83
Creating a Security Policy File	84
Edit a Security Policy File	85
Validating a Security Policy File	86
Creating a Security Token	87
Perform Additional Programming Tasks to Process Authentication Requests	87
Validating a Security Token File	88
Resetting the Validity for Policy, Token, or Domain Credentials	88
Resetting the Validity for a Policy when Data Encryption is Set	88
Manual Process of Changing the Security Policy and Token Files	89
Starting Security Domain Controllers	89
Starting a Security Domain Requestor with a Token File	90
ActiveSpaces Utilities as Windows Services	92
The as-service.tra File Example	92
The ActiveSpaces Hawk MicroAgent	94
Getting Started	94
ActiveSpaces Hawk Microagent Methods Overview	95
Space and Member Statistics Methods	95
Space Distribution Methods	96
Space Operation Throughput at Application Level Methods	96
Space Definition Events Methods	96
Metaspace Member and Space Member Events Methods	96
Console and File Logging Management Methods	97
Space Management Methods	97
ActiveSpaces Monitoring and Management	98
Starting ASMM	98
Setting up ASMM Credentials	99
ASMM Admin Home Page Reference	99
Creating a New Metaspace Connection	100
Connecting to a Metaspace	102
Viewing the Members and Spaces Connected to a Metaspace	102

Using the ASMM Terminal Window	103
Resetting the Statistics of a Space	103
Modifying a Metaspace Connection	103
Disconnecting a Member	104
Disconnecting from a Metaspace	104
Deleting a Metaspace Connection	104
Specifying Service Settings	105
Specifying Client Connection Settings	106
Cross-site Replication Concepts	107
Local Site Definition	108
Starting as-agent as the First Metaspace Member	109
Starting as-router as the First Metaspace Member	109
Starting a Custom Java Application as the First Metaspace Member	109
Remote Site Definition	110
The as-router Executable	112
Starting as-router	113
as-admin Commands for Cross-site Replication	115
Administering the Primary Site	115
Administering the DR Site	115
Deploying Cross-site Replication	115
Defining Spaces	115
Setting Up Cross-Site Replication on the DR Site	116
Setting Up Cross-Site Replication on the Primary Site	116
Monitoring Cross-site Replication	117
An Example of Simple Cross-site Replication	118
Shared-All Persistence Usage	118

Figures

Client-Server Deployment	11
Deployment with Shared-All Persistence	12
Deployment with Shared-Nothing Persistence	13
Sample Deployment with Host-Aware Replication	14
Remote Client Deployment	15
Concurrency with Remote Clients	16
Selecting an Interface and Network for Optimum Discovery	18
Specifying Multiple TCP Discovery Nodes	19
ASMM Admin Home Page	99
Metaspace Connection Dialog	101
Cross-site Replication Between Two Sites	108

TIBCO Documentation and Support Services

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the TIBCO Product Documentation website, mainly in HTML and PDF formats.

The TIBCO Product Documentation website is updated frequently and is more current than any other documentation included with the product. To access the latest documentation, visit <https://docs.tibco.com>.

Product-Specific Documentation

Documentation for TIBCO products is not bundled with the software. Instead, it is available on the TIBCO Documentation site. To directly access documentation for this product, double-click the following file:

`TIBCO_HOME/release_notes/TIB_activespaces_version_docinfo.html` where `TIBCO_HOME` is the top-level directory in which TIBCO products are installed. On Windows, the default `TIBCO_HOME` is `C:\tibco`. On UNIX systems, the default `TIBCO_HOME` is `/opt/tibco`.

The following documents for this product can be found on the TIBCO Documentation site:

Documentation for TIBCO ActiveSpaces® is available on the [TIBCO ActiveSpaces® Product Documentation](#) page.

- *TIBCO ActiveSpaces® Installation*
- *TIBCO ActiveSpaces® Administration*
- *TIBCO ActiveSpaces® Monitoring and Management Console Guide*
- *TIBCO ActiveSpaces® Developer's Guide*
- *TIBCO ActiveSpaces® C API Reference*
- *TIBCO ActiveSpaces® Java API Reference*
- *TIBCO ActiveSpaces® .NET API Reference*
- *TIBCO ActiveSpaces® Release Notes*

How to Contact TIBCO Support

You can contact TIBCO Support in the following ways:

- For an overview of TIBCO Support, visit <http://www.tibco.com/services/support>.
- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the TIBCO Support portal at <https://support.tibco.com>.
- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to <https://support.tibco.com>. If you do not have a user name, you can request one by clicking Register on the website.

How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the [TIBCO Ideas Portal](#). For a free registration, go to <https://community.tibco.com>.

Overview of Administration

TIBCO ActiveSpaces® provides an administrative API and two administrative tools: the Admin object provided in Java, .NET, and C, the as-admin command-line tool and the ActiveSpaces Monitoring and Management (ASMM) web-based tool.

as-admin

The main administration utility for ActiveSpaces. Provides a command line interface (CLI) that allows you to manage nodes in the ActiveSpaces data grid. Using as-admin, you can:

- Connect to a metaspace as a full peer or as a remote client.
- Connect to another member of a metaspace.
- Run administrative commands as a member of a metaspace or on other members of the metaspace.
- Run administrative commands that allow you to:
 - Display information about the metaspace and its members.
 - Export the metaspace.
 - Recover the metaspace or one of its spaces.
 - Manage metaspace transactions.
 - Display information about the spaces in the metaspace and their members, data, and locks.
 - Create, join, leave, and alter the definition of spaces in the metaspace.
 - Create and validate security policy files and security token files.
 - Change logging levels.

The Admin Object

An object provided in the Java, .NET, and C (tibasAdmin) APIs that provides a programmatic interface for executing most of the commands available in as-admin.

ASMM

ActiveSpaces Monitoring and Management (ASMM) provides a GUI that lets you connect to a metaspace, display information about various ActiveSpaces system resources, for example, metaspaces, metaspace members, spaces, entry statistics, and so on, and browse space entries.

The as-agent Process

In addition to the administration tools, ActiveSpaces provides a process called as-agent that is used:

- To facilitate deployment of the ActiveSpaces cluster by connecting to a metaspace, joining all distributed spaces in the specified metaspace as a seeder, and keeping the space active.
- Optionally, to run an as-admin console and issue as-admin administration commands.

The as-dump Utility

ActiveSpaces provides an as-dump utility that you can use to display information about shared-nothing persistence data files.

The as-router Process

ActiveSpaces provides a separate executable called **as-router** that is used in disaster recovery by replicating transactions and PUT and TAKE operations from an ActiveSpaces cluster on one site to a passive ActiveSpaces cluster on another site.

Deployment Modes

You can deploy in the following ways:

- Client-server deployment, also known as the leeches-seeders deployment.
- Shared-all persistence
- Shared-nothing persistence
- Host-aware replication
- Remote client
- Cross-site replication

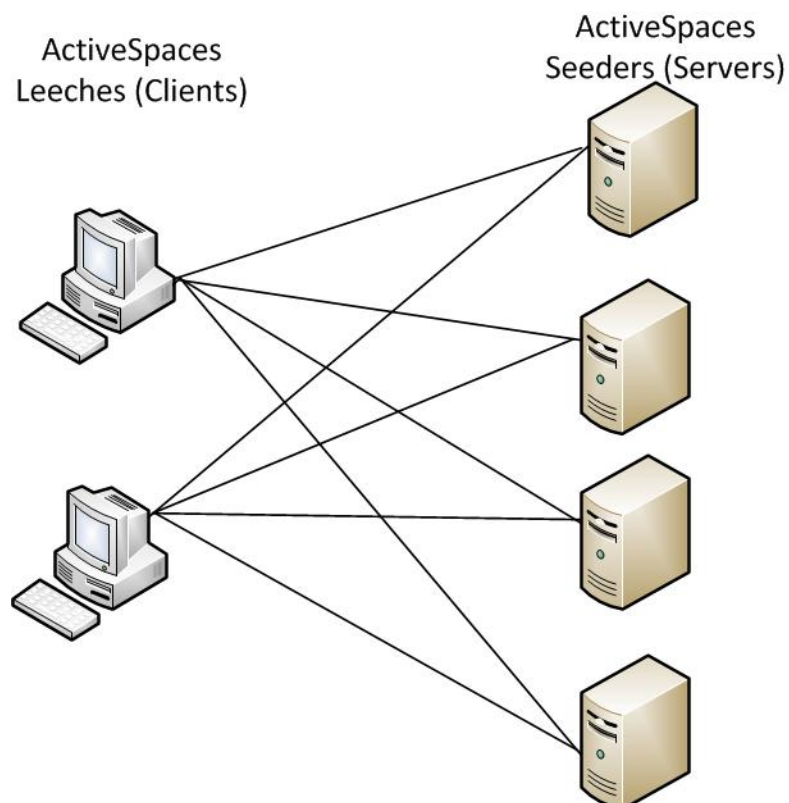
Clients (Leeches) and Servers (Seeders)

When a member joins a space, it can join as a server (seeder) or as a client (leech).

This allows you to distribute the node deployment between seeders and leeches:

- *Seeders* play an active role in maintaining the space by providing CPU and RAM.
- *Leeches* play a passive role. They have access to space data but provide no resources.

Client-Server Deployment



For more information on deploying seeders and leeches in the ActiveSpaces network, see *TIBCO ActiveSpaces Developer's Guide*.

Shared-All Persistence

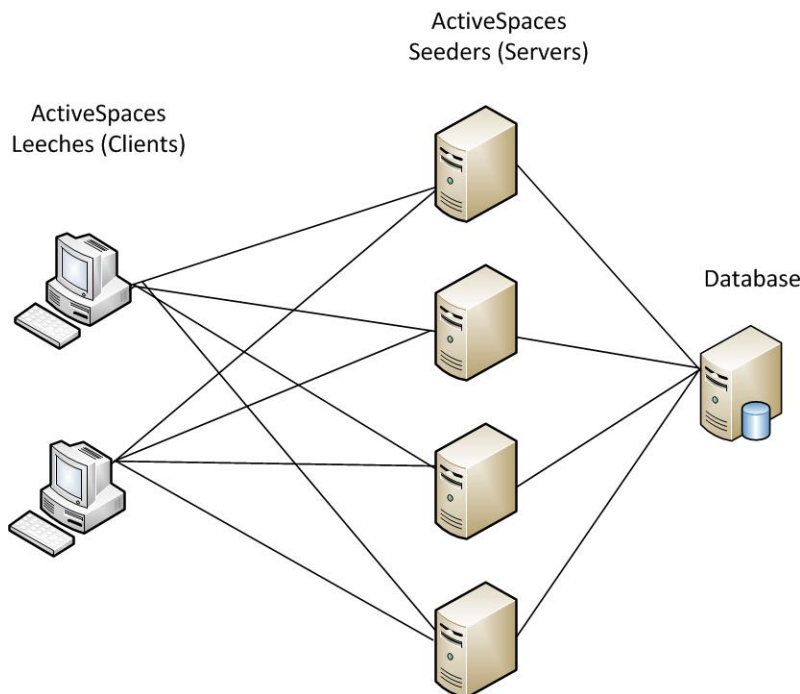
In ActiveSpaces, you can persist data to disk storage and recover it if data loss occurs or if there is a problem with the cluster startup.

For detailed information on persistence, see *TIBCO ActiveSpaces Developer's Guide*.

With shared-all persistence, all seeder nodes share a single persister or a set of persisters. Your application must provide an implementation of the persistence interface before it can interface with the shared persistence layer of choice.

[Figure 2, Deployment with Shared-All Persistence](#) shows all of the seeder nodes in an ActiveSpaces cluster sharing a single database for persistence and data recovery.

Deployment with Shared-All Persistence

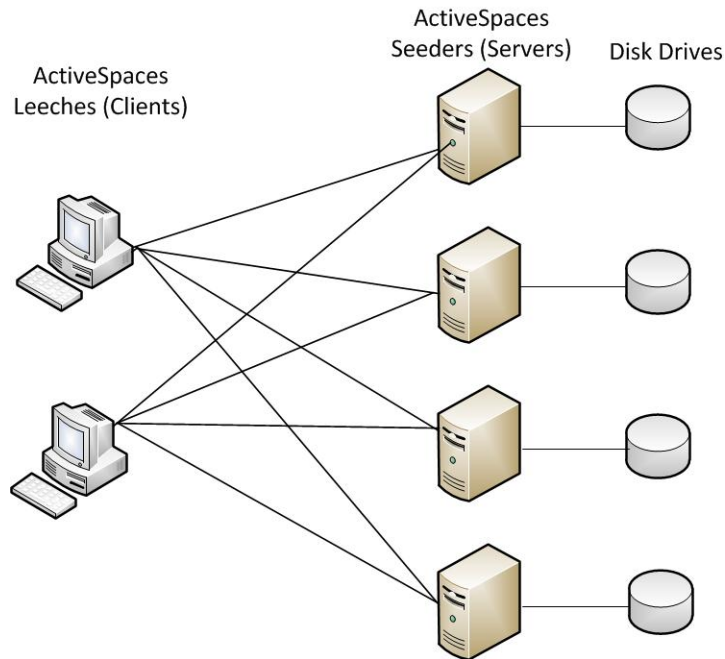


Shared-Nothing Persistence

With shared-nothing 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.

[Figure 3, Deployment with Shared-Nothing Persistence](#) shows a group of seeder nodes that persist data to local hard disk.

Deployment with Shared-Nothing Persistence



Host-Aware Replication

Use host-aware replication to ensure that the data for seeders running on the same machine is always replicated by seeders running on a different machine.

If there is only one seeder running on a machine, then host-aware replication is not needed, because data could not be replicated onto the same machine if there are no other seeders running on that machine.

To set up host-aware replication for the seeders on a machine, you logically group the seeders together using the following member naming convention:

`<group_name>.<member_name>`

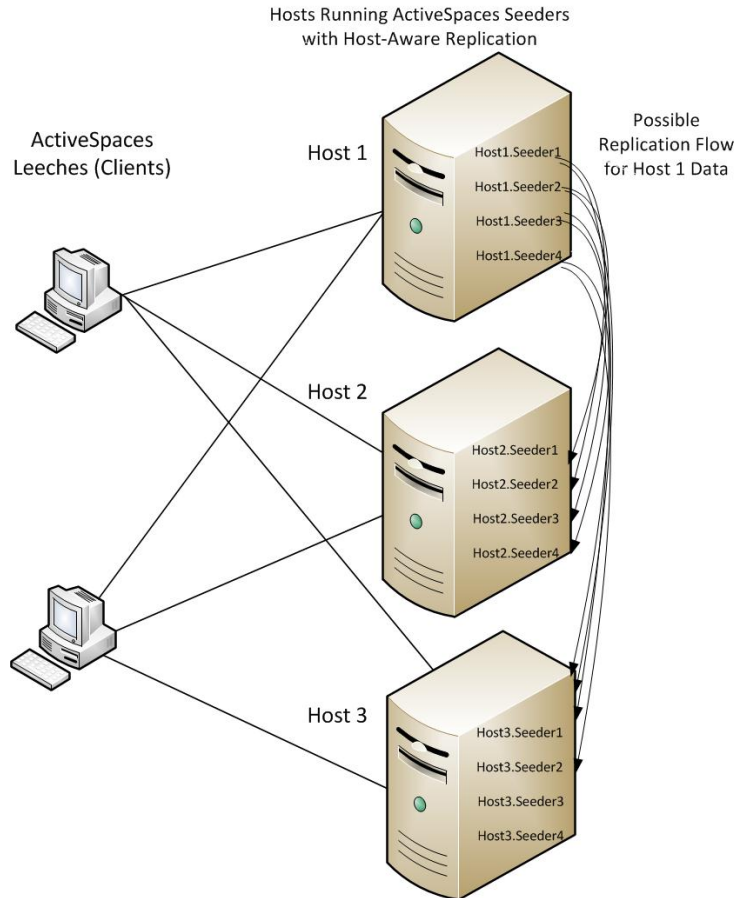
where *group_name* can be any name which helps you to identify the grouping of seeder members. For example, the *group_name* can be the name of the machine on which the seeders are running or some other arbitrary name that logically identifies the group of seeders.



Host-aware replication is purely based on the *group_name* part of the member name of the seeder, and not on the IP address of the physical host the process is running on. Ensure that the *<group_name>* or the *<member_name>* do not have embedded dots in them. For example, `Host . 1 . Seeder` is wrong. It should be `Host1 . Seeder`.

[Figure 4, Sample Deployment with Host-Aware Replication](#) shows a topology for host-aware replication in which each host is running four seeders and a seeder grouping is defined for each host.

Sample Deployment with Host-Aware Replication



Remember that when `replicate all` is set, the host aware settings are ignored and all seeders of the spaces get either the original copy or the replica.

The first host is named Host1, and runs four seeders, Seeder1, Seeder2, Seeder3, and Seeder4. The second host is named Host2, and the third host is named Host3. Each of these hosts also runs four seeders.

If you name the seeders according to host-aware replication member naming, then ActiveSpaces always replicates data that is seeded on one of the seeders on a given host to a seeder that is running on another host. Therefore, if a host or device goes down, the data that was seeded by any of the seeders on that host is not lost, because it is automatically backed up on a seeder that resides on another host.

You can set up this type of topology in several ways:

- By using the TIBCO ActiveSpaces API programs in an application program that implements `<group_name>.<member_name>` member names.

For information on the API functions to implement host-aware replication, see *TIBCO ActiveSpaces Developer's Guide*.

- By using `as-agent` and specifying the member names using the `-member_name` parameter.



The parameter, `-name` is deprecated.

For example, you might use `as-agent` with the `-member_name` parameter to start four seeders on each host. For example, on host 1, run `as-agent` as follows:

```
java -jar as-agent.jar -metaspace ms -discovery
tcp://127.0.0.1:50000 -listen tcp://127.0.0.1:50000
-member_name Host1.Seeder1
```

Then run two additional instances, with the `-member_name` parameter specifying `Host1.Seeder2`, and `Host1.Seeder3`.

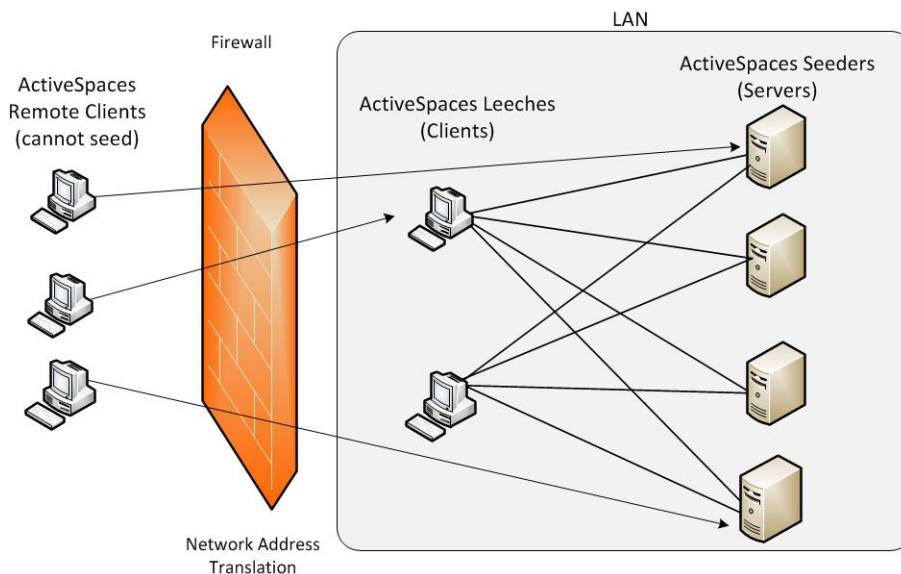
- When using `as-admin` to connect to a metaspace, the member name is specified using the `member_name` option of the `connect` command.

Remote Client Architecture

In some situations, where nodes are unable to become full peers in the data grid, for example, when there is a firewall protecting a LAN, you can deploy ActiveSpaces nodes as remote clients. Remote clients can perform puts and gets, but cannot act as seeders or assume a management role in the core cluster.

Using remote clients also has the advantage of saving cluster processing overhead.

Remote Client Deployment



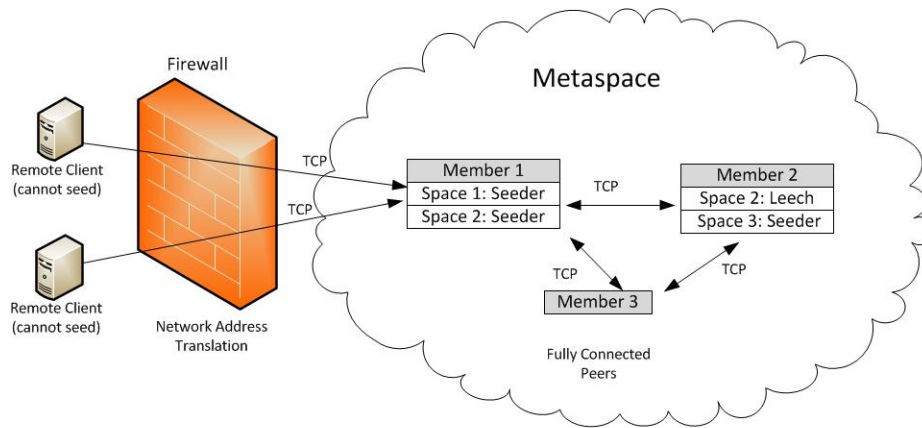
When you use remote clients, the remote clients can update the entries in the database, and ActiveSpaces seeders ensure that concurrency is maintained. Remote clients connect using TCP connections.



When a cluster restarts, ensure that you shut down and restart all cluster members. Ensure that you shut down and restart all remote clients (including Admin if it is connected as a remote client). Not doing so would result in the `REMOTE_CLIENT_TIMED_OUT` error. If you do not restart the remote clients, you cannot perform any operation on the cluster.

[Figure 6, Concurrency with Remote Clients](#), shows remote clients updating to a seeder (Member 1). The seeders in the metaspace ensure that the data replicated on Member 2 is synchronized with the data on Member 1.

Concurrency with Remote Clients



Concurrency at the Metaspace Level

Cross-Site Replication

Cross-site replication is used to dynamically backup changes to the data in your metaspace to another metaspace in a different geographical location.

For details, see the section [Cross-Site Replication Concepts](#).

Best Practices for Deploying in a Virtual Environment

The following best practices are applicable when you deploy ActiveSpaces in a virtual environment.

Disable vMotion

While migrating processes from one host to another, vMotion stops processes for a certain period. Consequently, you might lose some members, resulting in cluster instability. You might also experience delay in user operations.

Disable vSnapshot

While taking a consistent snapshot, vSnapshot briefly stops the processes for a certain amount of time which can negatively impact the performance of ActiveSpaces.

Pin the Virtual Machines

By pinning the resources, you can dedicate memory and CPU to an image. This reduces latency and risk of undue delays in ActiveSpaces. Furthermore, pinning helps you assign physical CPU cores and enables you to allocate NUMA aware memory.

Disable Memory Ballooning

Memory access speed and large memory allocation are key aspects of ActiveSpaces. Over allocating memory, as done by memory ballooning, hampers the speed and memory access patterns of ActiveSpaces.

Reserve Memory

By reserving memory, you ensure that after ActiveSpaces claims the physical pages, the Hypervisor cannot reclaim them with memory ballooning or Virtual Machine File Swapping techniques.

Best Practices for Node Discovery

Specifying discovery when using ActiveSpaces security, choosing the right discovery point, and specifying multiple TCP discovery nodes for fault tolerance are some approaches you can use for node discovery.

Specify Discovery When Using ActiveSpaces Security

If you are implementing ActiveSpaces security:

- The discovery transport type must be TCP.
- The discovery list on both the requestor and controller members for the given metaspace binding must be the same.

Also, when you implement ActiveSpaces security, you do not need to specify the discovery URL for a member that is joining the metaspace, because the discovery URL is retrieved from the security policy file or security token file used to start a node. If you do specify the discovery URL, it is ignored and ActiveSpaces uses the value for the metaspace as specified in the security configuration files.

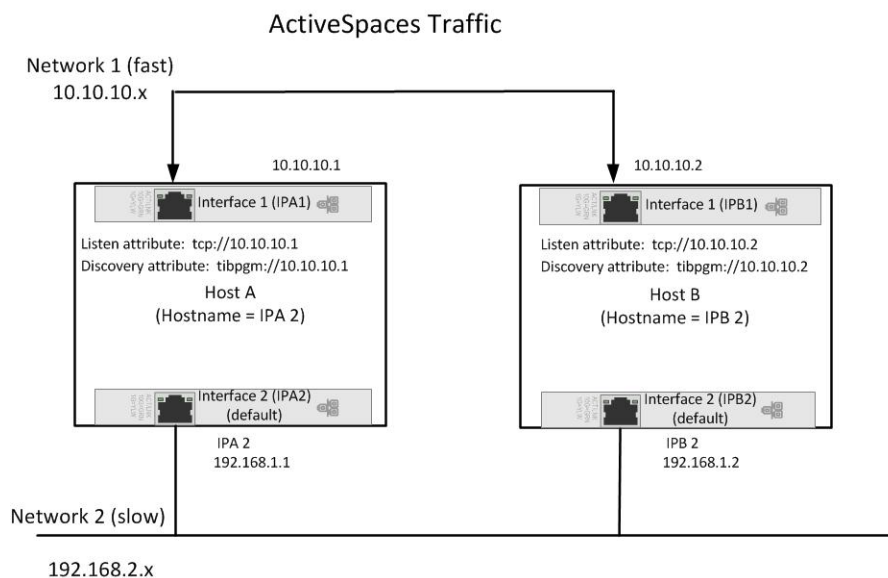
For more information on ActiveSpaces security, see *TIBCO ActiveSpaces Developer's Guide*.

Choose the Right Discovery Point

When you specify the discovery attribute (either in the discovery attribute of the MemberDef object for the space member or in the `discovery` parameter for the `as-admin connect` command), you can specify the IP address of the network interface to use for discovery.

It is important to specify the optimum discovery IP address. For example, in a network where you have several subnetworks, or where you have a relatively fast network and also a slower network, performance will be best if you choose the faster network. [Figure 7](#) shows a network topology that has a fast network and a slower network.

Selecting an Interface and Network for Optimum Discovery



In the network shown in [Figure 7](#), there are two networks: network 1 is fast and network 2 is slow.

To ensure that you use the interface connected to the faster network for discovery, specify the interface for the faster network. In the example, we specify the `tibpgm` discovery protocol and the IP address of the interface for the faster network:

```
tibpgm://10.10.10.1
```

This ensures that discovery uses the faster network.

Specification of Multiple TCP Discovery Nodes for Fault Tolerance

If you are using the TCP discovery method, you can specify multiple TCP discovery nodes to provide fault tolerance.

Figure 8, [Specifying Multiple TCP Discovery Nodes](#) illustrates how to specify multiple TCP discovery nodes in a network with four TCP hosts.

In the example, we specify the tcp discovery method and two node address/port pairs:

```
tcp://10.10.10.1:6000;10.10.10.2:6000
```

When setting up the discovery attribute in the MemberDef object for the space or with the **discovery** parameter for the **connect** command, you can specify additional IP addresses and ports, separated by semicolons.

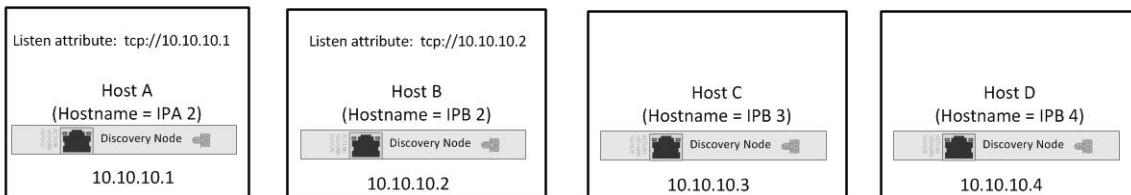
Specifying Multiple TCP Discovery Nodes

TCP Discovery

Host A and Host B are the designated discovery nodes

- Two discovery nodes are used for fault tolerance
- All are members of the same metaspace

Discovery attribute: tcp://10.10.10.1:6000;10.10.10.2:6000;



Admin CLI for ActiveSpaces Administration

Administrative tasks for TIBCO ActiveSpaces are performed through a utility called `as-admin`, or the *Administration Command Line Interface* (Admin CLI). These tasks include connecting to a metaspace, creating a space, and displaying information about existing spaces and members.

To start the TIBCO ActiveSpaces Admin CLI, you must:

- Set environment variables for TIBCO ActiveSpaces

For information on setting the required environment variables, see the *TIBCO ActiveSpaces Installation* document.

- Launch the Admin CLI, in a command prompt window

The executable for running the Admin CLI (`as-admin`) is located in the `/bin` directory of ActiveSpaces.



The Java version of `as-admin` is deprecated.

Additional Characteristics of the Admin CLI

- Arrow keys and the tab key function in the command window as in typical advanced shells.
- You can invoke a shell command by using the escape character `'!`, for example, `!dir` or `!ls` to list the files in the current directory.
- Field names and String literals must be enclosed in single or double quotes.
- In most cases, you will start using the Admin CLI by first using the **connect** command to connect to a metaspace.
- The default discovery mechanism is PGM. If RV discovery or TCP unicast discovery is needed, you must specify the appropriate discovery URL.

For more information on discovery, see the *TIBCO ActiveSpaces Developer's Guide*.

The Execute Method

Admin CLI administrative commands (for example defining a space) can be executed directly from within an application. This is done by using the Metaspace object's `execute` method and passing it a string representing the Admin CLI command. A string is returned containing the output resulting from executing the command.

Launching the Admin CLI

The default location of the Admin CLI program is:

```
<TIBCO_HOME>\as\<version>\bin
```

Procedure

1. Open a command window.
2. Make sure that you have set the required environment variables as described in the *TIBCO ActiveSpaces Installation* document.
3. Enter the following commands:

```
cd <TIBCO_HOME>\as\<version>\bin
as-admin
```

You can specify the following command line options when starting `as-admin`:

<code>[-help]</code>	display command line options
<code>[-i(nput) <filepath>]</code>	specify file which contains admin commands to execute
<code>[-s <start up command>]</code>	specify admin command to execute first
<code>[-c(onsole_log_level <log level>)]</code>	console logging level (fatal, error, warning, info, fine, finer, finest)
<code>[-f(ile_log_level) <log level>]</code>	enables file logging (fatal, error, warning, info, fine, finer, finest)
<code>[-l(og) <log file>]</code>	file to use for file logging, default is <current directory>/log/as-<pid>.log
<code>[-log_level <log level>]</code>	same as -file_log_level
<code>[-log_append (true false)]</code>	append to or clear contents of existing log file, default=true, append to
<code>[-log_count <integer>]</code>	maximum number of log files, default=1
<code>[-log_size <integer>]</code>	byte limit for log file, default=-1, unlimited
<code>[-monitor_system (true false)]</code>	enable or disable performance monitoring, default is false=disabled
<code>[-metaspace <string>]</code>	metaspace name, default=ms
<code>[-member_name <string>]</code>	unique member name, default=autogenerated name
<code>[-discovery <string>]</code>	discovery url, default=tibpgm
<code>[-listen <string>]</code>	listen url, default=tcp
<code>[-remote_listen <string>]</code>	remote listen url
<code>[-rx_buffer_size <string>]</code>	size of receive buffer to use for each transport connection, default 2mb
<code>[-worker_thread_count <integer>]</code>	number of threads to use for servicing remote invocations, default 32
<code>[-data_store <string>]</code>	data store path
<code>[-member_timeout <integer>]</code>	time in milliseconds to wait for a member to reconnect, default 30000
<code>[-cluster_suspend_threshold <integer>]</code>	lost hosts allowed before membership operations are suspended, default don't suspend
<code>[-connect_timeout <integer>]</code>	time (in ms) to wait to connect to metaspace, default 30000ms
<code>[-security_policy <string>]</code>	security policy path
<code>[-security_token <string>]</code>	security token path
<code>[-history_size <integer>]</code>	set size of command history buffer

The Admin CLI prompt appears:

```
as-admin>
```



The usage information for as-admin can be obtained by invoking `as-admin -help` from the ActiveSpaces bin directory in a command window.

You can now enter Admin CLI commands.

Using a Script File to Pass Arguments

Using a script file, you can pass commands for the Admin CLI to begin executing when you launch the Admin CLI.

```
as-admin -i admin-cmd-filename
```



Comments can be inserted into the script file by starting a line with #, as follows:

```
# this is a comment
```

This mechanism allows execution of commands in a batch mode through the Admin CLI. There are several limitations on the structure of the file:

- No special characters are allowed.
- Each line is executed sequentially by the Admin CLI. Therefore, each individual command must be entered on one line.
- The Admin CLI will not automatically exit and return to the shell after executing the commands of the file unless the script file contains a quit command.

The following is an example:

```
#####
connect name "UserMetaspace" discovery "tibpgm"
show members
export metaspace to "export.txt"
quit
#####
```

This will connect to a metaspace, display members, export the metaspace, and exit the Admin CLI .



The **export metaspace [to <filename>]** Admin CLI command generates a file containing the schema for the existing metaspace. For more details, see [export metaspace](#). The `admin-cmd-filename` value can be the path to this generated file. In this way, a new metaspace can be created using the exported metaspace schema.

Example

The following example demonstrates the following:

- invoking the Admin CLI using a script file to connect to a metaspace.
- creating a space.

Here is an example of the contents of the script file. In this example, the file is named `example01.txt` and is located in the directory `C:\temp2`.

```
connect name 'ms' discovery 'tibpgm' listen 'tcp'
define space name 'testspace' (field name 'key' type 'integer',
field name 'value' type 'string') key (type 'hash' fields ('key'))
```

Launch the Admin CLI using the following syntax:

```
as-admin -i c:\temp2\example01.txt
```

alter space

Use **alter space** command to add a field to an existing space definition, or to add or drop an index from a space definition.

Syntax


```
alter space <string> add (field name <string> type <string> [nullable <boolean>]
[encrypted <boolean>](, field name <string> type <string> [nullable <boolean>])*
alter space <string> add index ( name <string> [type <string>] fields (<string> (,
<string>)*))
alter space <string> drop index (<index_name> (, <index_name>)*)
```

Parameters

The following table describes the parameters for this command.

alter space Parameters

Parameter	Description
space	The name of the space to be modified.

Parameter	Description
add	<p>Specifies that a field is to be added:</p> <ul style="list-style-type: none"> • name specifies the name of the field to be added. • type specifies the data type for the field. Must be one of the following: boolean, char, short, integer, long, float, double, string, datetime, blob. • nullable specifies whether the field is nullable. The nullable value is optional. The default for nullable is true. • encrypted (optional) A boolean value specifying whether the field value is encrypted when it is stored in the space (default: false).
add index	<p>Specifies that an index is to be added:</p> <ul style="list-style-type: none"> • name specifies the name of the index. • type specifies the type of the index, which can be hash or tree (default: tree). • fields specifies a comma-separated list of field names to be used in the index. <p> When you add an index, the new index cannot have fields from an existing index.</p>
drop index	<p>Specifies that one or more indexes are to be removed.</p> <p>The <code>drop index</code> option must be followed by a comma-separated list of one or more index names specifying the indexes to remove.</p>

Examples

Examples for add field:

```
alter space "myspace" add (field name "average" type "double")
alter space "myspace" add (field name "average" type "double" nullable true)
alter space "myspace" add (field name "average" type "double", field name "total"
type "long" nullable true)
```

Examples for add index:

```
alter space add index (name "index1" type "hash" fields("a", "b", "c"))
alter space add index (name "index1" type "hash" fields("a", "b", "c")) index (name
"index2" type "hash" fields("a", "b", "c"))
```

Examples for drop index:

```
alter space drop index ("index1")
alter space drop index ("index1", "index2")
```

Example for adding a field and an index:

```
alter space "myspace" add (field name "average" type "double") index (name "index1"
type "hash" fields("a", "b", "c"))
```

The parameters for fields and index use the following format:

- **fields:** (field name "average" type "double")

- **index:** index (name "index1" type "hash" fields("a", "b", "c"))

You can perform consecutive updates by adding one field after another.

clear

The **clear** command is used to remove all lines of text from the command window except for a single Admin CLI prompt.

Syntax

```
clear
```

Parameters

None.

clear | set password

Use the **clear password** command to create a new policy file without a password or a new token file without a password. Use the **set password** command to create a new policy file or a new token file with a new password.

When you issue the **set password** command, you are prompted to enter and verify the new password for the domain.

Syntax


```
clear | set password domain_name <string> policy_file <string>
[new_policy_file <string>]
```

```
clear | set password token_file <string> [new_token_file <string>]
```

Parameters

The following table lists the parameters for this command with a description of each parameter.

clear | set password Parameters

Parameter	Description
domain_name	Specify the domain name for the domain for which you want to set or clear the password.
policy_file	Specify the existing policy file to use when creating the new policy file without a password.
new_policy_file	<p>This optional parameter specifies the name of a new policy file that is created based on the policy file that you specify with the <code>policy_file</code> parameter.</p> <p>If you do not specify a new policy filename, then a new policy file is created with the current date and time appended to the existing policy's filename. For example, if you specify an existing policy filename of "policy.txt," ActiveSpaces creates a policy with the name "policy.txt.2013_01_16_20_38_00."</p> <p> The policy filename cannot contain a forward slash character ("/").</p>

Parameter	Description
token_file	If you are using the clear password command to clear the password in a token file or the set password command to set the password for a token file, specify the token filename.
new_token_file	This optional parameter specifies the name of a new token file that is created based on the token file that you specify with the token_file parameter. If you do not specify a new token filename, then a new token file is created with the current date and time appended to the existing token's filename. For example, if the token file is named "my token," ActiveSpaces creates a token file with the name "mytoken.2013_01_16_20_38_00."

Example

```
clear password domain_name 'AS-DOMAIN' policy_file 'policy.txt'
new_policy_file 'newdeal.txt'
```



Parameter String values must be enclosed in either single or double quotes.

The following examples illustrate the syntax of the **clear password** command:

- `clear password domain_name 'AS-DOMAIN' policy_file 'policy.txt'`
- `clear password domain_name 'AS-DOMAIN' policy_file 'policy.txt' new_policy_file 'newdeal.txt'`
- `clear password token_file 'mytoken'`
- `clear password token_file 'mytoken' new_token_file 'yourtoken'`

close

The **close** command is used in conjunction with the **open** command which connects to a metaspaces through the listen port of another metaspaces member.

Any Admin CLI commands used while that connection is open is executed on the other metaspaces member. Use the **close** command to close the connection to the other metaspaces member's listen port. After using the **close** command, the Admin CLI will no longer be connected to a metaspaces, and you must issue another **open** command or a **connect** command to re-establish a metaspaces connection.

Syntax

```
close
```

Parameters

None

connect

connect is used to connect to a metaspaces. Connecting to a metaspaces is a necessary initializing step for the Admin CLI to begin working with ActiveSpaces, just as it is for an ActiveSpaces application. The Admin CLI can only be connected to one metaspaces at a time.

You cannot specify a token file and a policy file at the same time.


Syntax

```
connect [name <string>] [discovery <string>] [listen <string>]
[member_name <string>] [security_token <string>]
[security_policy <string>] [identity_password <string>]
[authentication_domain <string>]
[authentication_username <string>]
[authentication_password <string>]
[authentication_keyfile <string>]
```

Parameters

The table lists the parameters for this command with a description of each parameter.

connect Parameters

Parameter	Description
name	Optional. If a metaspace with this name does not exist, it will be created as a result of this command. If no name is specified, the Admin CLI will connect to the default metaspace, called <code>ms</code> .
discovery	<p>Optional. The discovery URL can take one of three forms:</p> <p>NOTE: If you are using ActiveSpaces security, you must use TCP discovery.</p> <ul style="list-style-type: none"> If Unicast discovery is used, then a list of 'well known' IP addresses and ports must be passed in a URL with the following syntax: <pre>tcp://ip1:port1;ip2:port2;...</pre> If multicast discovery is to be used then the URL must be as follows: <pre>tibpgm://destination port/ interface;discovery group address/optional transport arguments</pre> <p>See "PGM (Pragmatic General Multicast) URL Format" in the <i>TIBCO ActiveSpaces Developer's Guide</i> for more information on PGM discovery URLs.</p> <p> If you are connecting as a security domain controller or as a security domain requestor, do not specify the discovery parameter. If you do specify a discovery parameter, it will be overwritten by the discovery parameter specified in the security policy file or the security token file specified with the connect command.</p>

Parameter	Description
listen	<p>Optional. Specifies which interface and port the administrative process should create its listening TCP socket on.</p> <p>Syntax:</p> <pre>tcp://interface:port</pre> <p>See “Listen URL Format” in the <i>TIBCO ActiveSpaces Developer’s Guide</i> for more information on listen URLs.</p>
member_name	<p>Optional. Specifies a member name for the member. This helps to identify which member name is associated with which member ID. The show members command displays the member name if one has been assigned; otherwise, a default member name is assigned that is constructed from the member ID.</p>
security_token	<p>Optional. Specifies the token file for a security domain requestor that must be authenticated by a security domain controller.</p> <p>If TIBCO ActiveSpaces security is implemented and you are connecting from a requestor node, and the metaspace to which you are connecting requires a token file, specify the security_token parameter and provide the directory path and filename for the token file.</p> <p>If you specify a token file, do not specify the security_policy parameter.</p>
security_policy	<p>Optional. If TIBCO ActiveSpaces security is implemented and you are connecting from a domain security controller node, specify the security_policy parameter and provide the directory path and filename for the policy file.</p> <p>If you specify a policy file, do not specify the security_token parameter.</p>
identity_password	<p>Optional. A string containing the password for the identity key in the security policy file.</p>
authentication_domain	<p>Optional. A string containing the domain name for user authentication.</p>
authentication_username	<p>Optional. A string containing the username to authenticate.</p>
authentication_password	<p>Optional. A string containing the password for the username</p>
authentication_keyfile	<p>Optional. A string containing the full path for a file containing the key to use for authentication.</p>
member_timeout	<p>Specifies the time in milliseconds to wait for a member to reconnect. The default is 30000.</p>
cluster_suspend_threshold	<p>Specifies the lost hosts allowed before membership operations are suspended. By default, it is not suspended.</p>

Parameter	Description
connect_timeout	Specifies the time to wait to connect to the metaspacer.

Example

```
connect name 'ms' discovery 'tcp://192.168.1.10'
```



Parameter values must be enclosed in either single or double quotes.

The following examples illustrate the syntax of the **connect** command:

- connect
- connect name <metaspacer_name>
- connect discovery <discovery_url>
- connect listen <listen_url>
- connect discovery <discovery_url> listen <listen_url>
- connect name <metaspacer_name> discovery <discovery_url>
- connect name <metaspacer_name> discovery <discovery_url> listen <listen_url>
- connect name <metaspacer_name> discovery <discovery_url> listen <listen_url> member_name <member_name>
- connect security_policy 'mypolicy.txt' name 'ms' listen 'tcp://127.0.0.1:50000'
- connect security_token 'mytoken.txt' name 'ms' listen 'tcp://127.0.0.1:50000'

define | create safe_password

Passwords can be used in different contexts in ActiveSpaces. Depending on the configuration of the security domains, token files, and user authentication requirements, there are different options. The **define | create safe_password** command allows you to create safe passwords for different situations.

A new command is now available to generate safe passwords for different purposes.

Syntax

```
define | create safe_password for (identity | authentication)
```

Parameters

define | create safe_password Parameters

Parameter	Description
identity	Specify the safe password that is to be used to decrypt identities.
authentication	Specify the safe password that is to be used for user password authentication.

Remarks

If you choose to encrypt identities in domain or token files, you must use one of the following commands:

- `define | create security_policy ... encrypt true ... policy_file <string>`
- `define | create security_token ... create_identity ... encrypt true ... token_file <string>`



The default behavior forces the private key to be encrypted therefore if `encrypt true` is not used, the generated private key will be protected with the provided password

When safe passwords are created to be used to decrypt identities, use **`create safe_password for identity`**.

If client authentication is to be enforced in the security policy for a given cluster, then joining requestor members must provide a valid credential before being able to use cluster resources. If the authentication scheme is `userpwd` (`authentication=userpwd;...`) then the user must normally provide a username and a password (and an additional domain value if using system source and windows authentication where accounts reside on a central/corporate server).

When safe passwords are created to be used in this context, use **`create safe_password for authentication`**.

For both of the above cases, the command produces encoded passwords, which can only be used for the purpose created. The password can then be applied in command lines, scripts, APIs and even at password prompts.

Example:

```
as-admin> create safe_password for identity
Password: ...
Verifying - Password: ...
Safe password: #SAFE#e041rA3TWXxJmhiriab7wG1p+OQqDbxCI0dsrDhTcLdbM=
...
> as-examples -security_policy policy.txt -listen tcp://localhost -role seeder -
identity_password #SAFE#e041rA3TWXxJmhiriab7wG1p+OQqDbxCI0dsrDhTcLdbM=
as-admin> create safe_password for authentication
Password: ...
Verifying - Password: ...
Safe password: #SAFE#69+OgjeN0tWrlDkvpJQ6D/e81T3pUbLYhOoRH9dxKX/As=
```

define | create security_policy

Use the `define|create security_policy` command to create a security policy file.



Syntax



```
define | create security_policy [policy_name <string>]
[encrypt <boolean>][validity_days <integer>] policy_file <string>
```

Parameters

The following table lists the parameters for this command with a description of each parameter.

define | create security_policy Parameters

Parameter	Description
policy_name	<p>Optional. Specifies the name of the policy to be created. If you do not specify a policy name, the policy is given the default name AS-POLICY.</p> <p> You cannot specify a policy file and a security token for the same connection.</p> <p>You can also specify one or more domains that the policy is associated with:</p> <p>To specify that the policy is associated with one domain, specify the policy name and the domain as follows:</p> <pre>define create security_policy policy_name <policy_name>/ <domain name> policy_file <string>.</pre> <p>For example:</p> <pre>create security_policy policy_name "OUR_POLICY/OUR_DOMAIN" policy_file "ourpolicy.txt"</pre> <p>If you enter the command in this way, the encrypt setting defaults to false; then if you specify one domain, you are prompted to enter and verify the password for that domain. If you specify multiple domains, you are prompted to enter and verify the password for each domain.</p> <p> If you specify encrypt=false, ActiveSpaces creates all domains is created with an unencrypted ID, which requires no password, and you are not prompted for a password.</p> <p>To create multiple domains associated with the policy, specify the policy name and a list of domains that the policy is associated with. Specify the domains separated by commas:</p> <pre>define create security_policy policy_name "<string/string, string, string ...>" policy_file <string></pre> <p>For example:</p> <pre>create security_policy policy_name "NEW_POLICY/MD1,MD2,MD3" policy_file "newpolicy.txt"</pre>
encrypt	<p>Optional. Indicates whether the private key for the policy is to be encrypted. The default is encrypt true.</p> <p>If you specify encryption, as-admin prompts you to specify and verify a new domain password and creates an encrypted private key in the Domain Identity section of the policy file.</p> <p>If you specify encrypt false, the domain does not require a password, and as-admin creates an unencrypted private key in the policy file.</p>
validity_days	<p>An integer that specifies how long the domain ID that the command creates remains valid. The default value is 365 days.</p> <p>Policies can have more than one domain, where (in theory) each of them can have different validity days if the domain definitions are moved between policy files manually.</p>

Parameter	Description
policy_file	Enter the name of the policy file that is to be created for the policy.
	 You cannot specify a policy file and a security token for the same connection.
	 The policy filename cannot contain a forward slash character ("/").

Example

The following examples illustrate the syntax of the `define | create security_policy` command:

- `create security_policy policy_name 'mypolicy' policy_file 'policy.txt'`
- `create security_policy policy_name 'mypolicy' encrypt false policy_file 'policy.txt'`
- `define security_policy "MY_POLICY/MY_DOMAIN" policy_file 'policy.txt'`



Parameter values must be enclosed in either single or double quotes.

define | create security_token

Use the `define | create security_token` command to create a security token for deployment to ActiveSpaces requestor nodes.

When you enter the command, you are prompted to enter and verify a new token password for the security token. Enter and verify the password.

Syntax

```
define | create security_token domain_name <string>
policy_file <string> [create_identity [common_name <string>]
[encrypt <boolean>][validity_days <integer>]]token_file <string>
```

Parameters

The following table lists the parameters for this command with a description of each parameter.

define | create security_token Parameters

Parameter	Description
domain_name	Specifies the name of the domain for which the security token is to be created.
policy_file	Specifies the name of the policy file that is to be used to create the token.
create_id	Optional. Enter the <code>create_id</code> parameter if you want to create a private key to verify the identity of connecting nodes.

Parameter	Description
common_name	Optional. If you enter the create_id parameter and you want to provide an X.509 common name to identify the private key, specify a common name. If you do not specify a common name, ActiveSpaces generates a common name that contains the domain name plus a random number; for example "/CN=AS-REQUESTOR-FEF3A467." If there is no common name associated with the token, then node connections use a temporary name generated by ActiveSpaces. If you provide a common name for the token file, this name is always used.
encrypt	Optional. If you enter the create_id parameter and you want to encrypt the private key, enter encrypt true (the default setting). If you do not want to encrypt the private key, enter encrypt false. Using encrypt false eliminates having to enter the password each time the node is started.
validity_days	Optional. To specify the number of days that the private key is valid for, enter the number of days. The default setting is 365 days.
token_file	Provide the name of the token file that is to be created.

Example

The following examples illustrate the syntax of the **define | create security_token** command:

- `create security_token domain_name 'AS-DOMAIN' policy_file 'policy.txt' create_identity common_name 'MyRequestor-123' encrypt true validity_days 90 token_file 'mytoken'`
- `create security_token domain_name 'AS-DOMAIN' policy_file 'policy.txt' create_identity token_file 'newtoken'`
- `create security_token domain_name 'AS-DOMAIN' policy_file 'policy.txt' create_identity common_name 'MyRequestor-123' encrypt true validity_days 100 token_file 'mysecurity_token'`



Parameter values must be enclosed in either single or double quotes.

define | create space

Used to create a space.

Syntax

```
define | create space <string>
(field name <string> type <string> [nullable <boolean>] (, field name <string> type
<string> [nullable <boolean>] [encrypted <boolean>]*)
  key ( [type <string>] fields (<string> (, <string>)*))
  distribution_def ('KEY','field0','field1','field2' ...)
  (index ( name <string> [type <string>] fields (<string>
(, <string>)*)))
[distribution_fields (<string> (, <string>)* )
[distribution_policy <string>]
[replication_count <integer>] [replication_policy <string>]
[host_aware_replication <boolean>]
[persistence_type <string>] [persistence_policy <string>]
[file_sync_interval <long>]
[cache_policy <string>]
```



```
[min_seeders <integer>]
[capacity <long>] [eviction_policy <string>]
[ttl <long>] [lock_ttl <long>]
[lock_wait <long>] [lock_scope <string>]
[space_wait <long>]
[write_timeout <long>] [read_timeout <long>]
[query_timeout <long>] [query_limit <long>]
[forget_old_value <boolean>]
[virtual_node_count <integer>]
[routed <boolean>]
```

Remarks

The supported data types for fields are:

Field types

boolean, char, short, integer, long, float, double, blob, string, datetime

Distribution policies

non_distributed, distributed

Persistence types

none, share_all, share_nothing

Persistence policies

sync, async

Replication policies

sync, async

Eviction policies

none, lru

Lock scopes

thread, process



Parameters

The parameters for this command are listed and described in [define create space Parameters](#).

define | create space Parameters

Parameter	Description
space	Required. Specify the name of the space that is to be created.
field	Required. The data type for a field must be one of the following: boolean, char, short, integer, long, float, double, string, datetime, blob.
nullable	Optional. Can be either true or false (no quotes). By default is equal to false. If a field has nullable set to true, tuples put into the space do not need to contain a field with that name.

Parameter	Description
encrypted	<p>Optional. If the field is not a key field or an index field and you have enabled ActiveSpaces security, you can specify that the data in the field is encrypted. Each (non-key, non-index) field can be made encrypted, as long as the policy for the corresponding domain allows it.</p>
key	<p>Required. Identifies one or more fields (already specified with the field parameter) that will serve as a unique key for the space.</p> <p>When you enter the key parameter, you can optionally specify the index type of the key field by including the type keyword. For example:</p> <pre>key (type "hash" fields (...))</pre> <p>The fields keyword is required.</p> <p>The type keyword is optional. The default index type is the "hash" index type.</p>
index	<p>Optional. Identifies one or more fields already specified with the "field" parameter that will serve as a secondary index. You can specify an index name and index type to be used by entering:</p> <pre>index (name "index1" type "tree" fields (...))</pre> <p>The name keyword is required.</p> <p>The type keyword is optional. The default is index type is "tree."</p> <p>The fields keyword and fields are required</p> <p>You can specify as many indexes as desired by specifying the indexes, one by one, after the key parameter. You just need to put them one after other after the key field. For example:</p> <pre>key (...) index(name "index1" ...) index(name "index2" ...) index (name "index3" ...)</pre>

Parameter	Description
distribution_fields	<p>Defines one or more fields as distribution fields. If a field is defined as a distribution field, then all tuples that have an identical data value for the field are stored on the same seeder.</p> <p>The distribution fields must be a subset of the key fields. Otherwise, ActiveSpaces throws an exception.</p> <p>The following example shows how to set up distribution fields:</p> <pre>key (fields ('KEY','field0','field1','field2')) distribution_fields ('KEY','field0','field1','field2') distribution_policy 'distributed' replication_count 0</pre> <p>In the example, field0, field1, and field2 are defined as key fields and also as distribution fields.</p> <p> If you define fields as distribution fields, then you must also set distribution_policy to distributed.</p>
distribution_policy	<p>Optional. Determines whether management of entries in the space is shared among the seeders that have joined the space (distributed) or a single seeder is responsible for all entries in the space (non_distributed). The default value is distributed.</p> <p> If you define fields as distribution def fields, then you must also set distribution_policy to distributed.</p>
replication_count	<p>Optional. An integer that specifies the number of times each entry should be replicated on different seeders (default: 0).</p>
replication_policy	<p>Optional. A value of sync specifies that replication is done in synchronous mode for the space, before the operation returns. When an operation modifies one of the entries in the space, the operation only returns an indication of success when that modification has been positively replicated up to the degree of replication required for the space. A value of async specifies that replication is asynchronous.</p>
host_aware_replication	<p>Optional. A boolean indicating whether host aware replication is enabled or disabled. Host aware replication ensures that data is not replicated on any seeders specified to be in the same seeder group (default: true, host aware replication is enabled).</p>

Parameter	Description
persistence_type	<p>Optional. Specifies whether persistence is enabled for the space, and if so, what type of persistence to use.</p> <p>To specify no persistence, specify none. To specify shared all persistence (space members designated as persisters maintain data on disk), specify shared-all. To specify shared-nothing persistence (each member maintains data on disk), specify share_nothing.</p>
persistence_policy	<p>Optional. Specifies the what type of communication is used to maintain persistence: synchronous (sync) or asynchronous (async).</p>
file_sync_interval	<p>A long integer that indicates the amount of time (in milliseconds) to wait between persists to the data store when asynchronous, versus shared-nothing persistence is used (default: 10000).</p>
min_seeders	<p>Optional. Specifies the minimum number of seeders that should be joined to the space before the space becomes ready to accept operations. The default value is 1.</p>
capacity	<p>Optional. Specifies a maximum number of entries per seeder for the space. When the capacity is reached the result of any additional request to put (insert) a new entry in the space will depend on the value of the <code>eviction_policy</code> attribute. The default value is -1 (no capacity).</p>
eviction_policy	<p>Optional. If a put operation on a space would cause a seeder to exceed the space's <code>capacity</code> attribute, then the value of this attribute will dictate the result of this operation: if the value is 'none' (in quotes) then there will be no eviction and the operation will fail because the seeder is already at capacity. If the value is 'lru' (in quotes) then the seeder will evict another entry from the space using the 'least recently used' eviction algorithm. The default value is 'none' (no eviction).</p>
ttl	<p>Optional. Time to live in milliseconds. The default is -1 (forever).</p>
lock_ttl	<p>Optional. Specifies in milliseconds the duration of a lock placed on the space. The default is -1 (forever).</p>
lock_wait	<p>Optional. For a space that is locked, specifies how long a member process will wait for it to become unlocked. The default is 0. Other accepted values are only positive values. The unit of measure is milliseconds.</p>

Parameter	Description
lock_scope	<p>Optional. Specifies the lock scope to be used for each operation that includes locking.</p> <p>You can specify the following:</p> <ul style="list-style-type: none"> • <code>thread</code> The lock applies to the current thread only. • <code>process</code> The lock applies to the entire application. <p>The default value is <code>thread</code>.</p>
space_wait	<p>Specifies the space wait for the specified space.</p> <p>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.</p>
write_timeout	<p>A long integer indicating the amount of time (in milliseconds) a write operation on the space can be blocked from completing the write before failing (default: 60000.)</p>
query_timeout	<p>A long integer indicating the amount of time (in milliseconds) a query on the space can take to return its results (default: -1, take forever).</p>
query_limit	<p>A long integer specifying the maximum number of entries to be returned by a query on this space (default 10000). The <code>query_limit</code> setting is intended to help you prevent large queries from exhausting the system's memory. By default, all browsers and listeners use the <code>query_limit</code> defined for the space they are browsing or listening on.</p> <div data-bbox="874 1224 1469 1766" style="border-left: 1px solid #ccc; padding-left: 10px; margin-left: 20px;"> <p>When there are multiple seeders for a space, the query limit is divided evenly amongst the seeders of the space so that an even number of resulting entries is taken from each seeder. When the number of entries in the result of a query approaches the query limit, it is possible that some seeders might return fewer entries than others. This is because the algorithm ActiveSpaces uses to distribute entries amongst seeders does not guarantee the entries will be evenly distributed. You should adjust the <code>query_limit</code> setting to accommodate the largest number of entries you will allow queries to return. When doing so, keep in mind that the <code>query_limit</code> should be slightly larger than the intended amount to account for the uneven distribution of entries amongst seeders.</p> </div>
read_timeout	<p>A long integer indicating the amount of time (in milliseconds) a read operation on the space can be blocked from completing the read before failing. (default: 60000).</p>

Parameter	Description
forget_old_value	A boolean value indicating whether operations that update entries in the space return any existing values for the entries (default: false, values are returned).
virtual_node_count	An integer that represents the capacity of each seeder for consistent hashing. (default: 1000).
routed	A boolean value indicating whether updates to the space can be routed between sites (default: false).

Examples

Simple example

```
define space 'myspace' (field name 'key' type 'integer'
field name 'value' type 'string') key (fields ('key'))
```

With additional parameters

```
define space 'testspace' (field name 'key' type 'double'
field name 'value' type 'blob') key (fields ('key'))
distribution_policy 'non_distributed' replication_count 1 replicated_policy 'sync'
capacity 10000 eviction_policy 'none' persistence_type 'share_nothing'
persistence_policy 'sync'
```

With distribution key parameters

```
define space 'usertable3' (field name 'KEY' type 'string',
field name 'field0' type 'string', field name 'field1' type 'string',
field name 'field2' type 'string', field name 'field3' type 'string',
field name 'field4' type 'string', field name 'field5' type 'string',
field name 'field6' type 'string', field name 'field7' type 'string',
field name 'field8' type 'string', field name 'field9' type 'string')
key (fields ('KEY', 'field0', 'field1', 'field2'))
distribution_def ('KEY', 'field0', 'field1', 'field2')
distribution_policy 'distributed' replication_count 0
```

disconnect

Disconnects the invoking metaspace member from the metaspace to which it is currently connected.

Syntax

```
disconnect
```

Parameters

None

drop space

Drops the named space.

Syntax

```
drop space <name>
```

Remarks

It is required that no members be joined to the space when the drop space command is executed.



Note that by definition, the as-agent process joins all user spaces as a seeder. the drop space command might fail if you are running as-agent in the same cluster.

Parameters

The following table shows parameters for this command.

drop space Parameters

Parameter	Description
name	Name of the space to be dropped.

execute on member

You can use the **execute on member** command to cause the execution of a specified Admin CLI command on a specified member of the metaspace. The results of the command are returned to the member that originally invoked the **execute on member** command.

Syntax

```
execute on member <string> <admin_command>
```

Parameters

The following table describes the parameters for this command.

execute on member Parameters

Parameter	Description
member	Specifies the metaspace member to execute an Admin CLI command on.
admin_command	Specifies an Admin CLI command to execute.

Example

```
execute on member "member1" show space "space_five"
```

execute on members

You can use the **execute on members** command to execute a specified Admin CLI command on all of the members in the metaspace, all peer members, or all remote client members.

Syntax

```
execute on [peer | remote] members <admin_command>
```

Parameters

The following table describes the parameters for this command.

execute on members Parameters

Parameter	Description
admin_command	Specifies an Admin CLI command to execute.

Example

```
execute on members show space "myspace"
```

add proxy

The **add proxy** command is used to add a proxy URL to the discovery list specified. This command is applicable only on remote clients. This command is used to dynamically maintain the list of proxies for remote clients without having to restart the remote client members.

Syntax

```
add proxy <string>
```

Parameter

The parameter for this command is listed in the following table:

Property	Description
IP_Address:Port	Specify the proxy URL. This parameter is a String in the form of IP address and a port separated by a colon, for example, 127.0.0.1:8080.

execute on proxy (deprecated)

Use the **execute on proxy** command to execute a specified Admin CLI command on the metaspace member acting as a proxy for the invoking member. This is when the invoking member is connected to a metaspace as a remote client.

When as-admin is connected as a remote client to a metaspace, you can use the **execute on proxy** command to cause the specified Admin CLI command to be run on the metaspace member acting as a proxy for the remote client instead of specifying the proxy member's name in the execute on member command.

Syntax

```
execute on proxy <admin_command>
```

Parameters

The following table describes the parameters for this command.

execute on proxy Parameters

Parameter	Description
admin_command	Specifies an Admin CLI command to execute.

drop proxy

The **drop proxy** command is used to drop a proxy URL from the discovery list specified. This command is applicable only on remote clients. This command is used to dynamically maintain the list of proxies for remote clients without having to restart the remote client members.

Syntax

```
drop proxy <string>
```

Parameter

The parameter for this command is listed in the following table:

Property	Description
IP_Address:Port	Specify the proxy URL. This parameter is a String in the form of IP address and a port separated by a colon. For example 127.0.0.1:8080.

export metaspaces

Exports the definitions for the metaspaces and its spaces (if any are currently defined) to a designated text file or to the screen.

Syntax

```
export metaspaces [to <filepath>]
```

Remarks

The file that is created can be used as a parameter when invoking as-admin with the **-i** parameter, as shown below.

Example Using as-admin

Export a metaspaces to a file by invoking export metaspaces, including 'to' and a filepath parameter. Use single- or double-quotes around the filepath:

```
as-admin> export metaspaces to 'C:\temp3\saved_metaspaces.txt'
```

Quit as-admin:

```
as-admin> quit
```

Launch as-admin with the **-i** flag and the filepath (no quotation marks):

```
<TIBCO_HOME>\as\<version>\bin>as-admin. -i  
C:\temp3\saved_metaspaces.txt
```

The output will indicate that the metaspaces has been recreated and as-admin is now connected to this newly-created metaspaces:

```
Connected to metaspaces ms
```

Parameters

The table shows parameters for this command.

export metaspace Parameters

Parameter	Description
filepath	Optional. The path to a location where the text file will be saved. If no filepath is given, the metaspace information is displayed on the screen.

help

Provides a complete list of Admin CLI commands and their syntax.

Syntax

```
help
help [-long] command_name
help [-long] command_name subcommand_name
```

Parameters*help Parameters*

Parameter	Description
none	Without any parameters, the full list of Admin CLI commands and their syntax is displayed.
-long	Includes parameter information along with the syntax and description of individual commands.
command_name	Specifies the command to show help information for. When there are several commands which start with the specified command_name, a list of those commands is displayed and use of the -long parameter has no effect.
subcommand_name	The command_name along with the subcommand_name is used to identify an individual command when there are multiple commands which start with command_name. For example, ' help show space ' or ' help show members '.

join space

Use the **join space** command to cause the as-admin metaspace member to join an existing space. If the space name does not exist, an error is returned.

The invoking metaspace member joins the space as a leech.

Syntax

```
join space <string> [role <string>]
```

Parameters

The following table describes the parameters for this command.

join space Parameters

Parameter	Description
space	Specifies the space to join.
role	This is optional. This parameter takes a String indicating the role of the member. Can be either a seeder or leech. Default role: leech.

leave space

Use the **leave space** command to cause the as-admin metaspace member to leave a space. An error is displayed if as-admin is not already a member of the space.

If you issue the **leave space** command and ActiveSpaces routing is enabled, then you can use the command to cause another metaspace member to leave a space.

Syntax

```
leave space <string>
```

Parameters

The following table describes the parameters for this command.

leave space Parameters

Parameter	Description
space	Specifies the name of the space to leave.

open

The **open** command is used to connect to a metaspace through the listen port of another metaspace member. Any Admin CLI commands issued while that connection is open will be executed on the other metaspace member. Use the **close** command to close this type of metaspace connection.

Syntax

```
open name <string> listen <string>
```

Parameters

The following table describes the parameters for this command.

open Parameters

Parameter	Description
name	Specifies the name of the metaspace to connect to.
listen	Specifies the listen URL of the metaspace member to use for connecting to the metaspace. Has the format <code>tcp://<IP address>:<port></code> .

Remarks

Note the following points regarding the **open** command:

- After you have opened a connection to the listen port of a metaspace member, *all* Admin commands issued from `as-admin` are then sent to the metaspace member to be executed. This includes commands such as `execute on member`.
- If you are already connected to a metaspace, you are not allowed to open a connection to the listen port of a metaspace member. You must first disconnect from the metaspace and then open the connection. Otherwise, you will receive an error when you try to use the **open** command.
- If you have opened a connection to the listen port of a metaspace member, you cannot directly connect to a metaspace using the **connect** command. You must first close the connection to the metaspace listen port and then connect to the metaspace. Otherwise, you will receive an error when you try to run the **connect** command.
- Only `as-admin` can be used to connect to a metaspace through another member's listen port.
- The `open` and `close` commands are not available through the Admin API.
- The `open` and `close` commands are not available when `as-agent -admin` is invoked, because an `as-agent` is already connected to a metaspace and having it disconnect is not the purpose of an `as-agent`.
- From `as-admin` you can repeatedly issue `connect/disconnect` and `open/close` commands. For example, you can connect to a metaspace, run some Admin commands, disconnect from the metaspace, open a connection to a metaspace member, run some Admin commands, close the connection, open a connection to another metaspace member, run some Admin commands, close the connection, connect to a metaspace, run some Admin commands, disconnect from the metaspace.
- Connecting to a metaspace through a metaspace member's listen port is not supported when using security for opening the connection is required.
- When the `open` command is run and a connection to the listen port is established, the `show metaspaces` command is run by the `open` command to ensure the connection is valid and to allow the user to see the metaspace information, which includes the listen port of the connected member. In this way the user can verify that they have connected to the desired metaspace member.

quit | exit | bye

Exits the Admin CLI.

Syntax

```
quit | exit | bye
```

Parameters

None.

recover metaspace

Used to restore all of the spaces of a metaspace when shared-nothing persistence is used.

Syntax



The options `with data` and `without data with recovery_policy`, are mutually exclusive. You can only use one of them in the command.

```
recover metaspace
    [recovery_policy <string>
    | with data (deprecated)
```

```
| without data (deprecated)]
[quorum <integer>]
[keep_files <boolean>]
[thread_count <integer>]
```



The options with `data` and `without data` with `recovery_policy`, are mutually exclusive. You can only use one of them in the command.

Parameters

The following table lists the parameters for this command with a description of each parameter.

recover metaspace Parameters

Parameter	Description
recovery_policy	<p>Optional. A string indicating which of the following recovery policies to use when restoring the data for each space in the metaspace:</p> <ul style="list-style-type: none"> • <code>no_data</code> - Does not recover data with the metaspace. • <code>without_data</code> - Deprecated. Use <code>no_data</code> instead. Specifies that the spaces of the metaspace should be restored without their data. • <code>no_data_loss</code> - (Default.) Start data recovery when there are enough seeders to ensure data loss does not occur. • <code>with data</code> - (Optional/Deprecated). Use <code>no_data_loss</code> instead. Specifies that the spaces of the metaspace should be restored with their data which was previously persisted to disk. 'with data' is the same as specifying <code>recovery_policy no_data_loss</code>. • <code>data_loss</code> - Start data recovery immediately even if there is a chance of a data loss. • <code>fast_load_only</code> - Start data recovery once all seeders are ready. • <code>robust_load_only</code> - Start data recovery when there are enough seeders to ensure data loss will not occur. This option will ensure fast load is not attempted. • <code>force_load</code> - Start data recovery immediately even if data loss will occur. Ignores all checks for required hosts/seeders and fast load is not attempted.
data	<p>Optional/Deprecated. Specifies whether the data from the space should be restored with the space. Specifying <code>with</code> indicates that the data from the space, which has been persisted to disk, should be restored with the space. Specifying <code>without</code> indicates the space should be restored without its data.</p> <p>The options with <code>data</code> and <code>without data</code> are now deprecated. Use <code>no_data_loss</code> and <code>no_data</code> instead respectively.</p>

Parameter	Description
quorum	Optional. An integer that specifies the minimum number of seeders which are needed for recovery to start. The default is 1.
keep_files	Optional. A boolean value that indicates whether the persistence files should be saved or deleted after recovery is complete. The default is <code>false</code> .
thread_count	Optional. An integer that specifies the number of threads to use for recovering spaces during the recovery process. Spaces can be recovered in parallel using this option. The default is 1.



When recovering a space which has a capacity defined, by default only the keys and indexes of the space are restored to memory regardless of whether or not the capacity has been reached. When capacity has been defined for a space, you can force the recovery of all data for a space, up to the capacity, by using the `robust_load_only` recovery policy. For any other recovery policy setting, the default behavior of only loading keys and indexes to memory is enforced. The data part of the tuple gets loaded into memory on subsequent access.

Example

```
recover space 'myspace' no_data_loss
```



String values passed as parameters must be enclosed in either single or double quotes.

recover space

The **recover space** command is used when all of the members of a metaspace have stopped and you are restarting those members. The recover space command allows you to control how persisted data for an individual space is restored.

Syntax



The options with `data` and without `data` with `recovery_policy`, are mutually exclusive. You can only use one of them in the command.

```
recover space <string>
  [recovery_policy <string> | with data | without data]
  [quorum <integer>]
  [keep_files <boolean>]
```

In case you need to restart the members of a metaspace and you have a space defined to persist data to a datastore, before using the space, at startup, ensure that you run the **recover space** command. If you do not run the command at startup, spaces with data in the datastore continue to be in the recover state not in the ready state.

Parameters

The following table lists the parameters for this command with a description of each parameter.

recover space Parameters

Parameter	Description
space	The name of the space to recover.
recovery_policy	<p>Optional. A string containing one of the following:</p> <ul style="list-style-type: none"> no_data - Does not recover data with the space. <p>When you recover a space without data, the persistence files are not deleted from the datastore. This is true even when you set <code>keep_files</code> to be <code>false</code>. For example, when you issue the command <code>recover space "<space_name>" no_data keep_files false</code>, the persistence files are not deleted from the datastore.</p> <ul style="list-style-type: none"> no_data_loss - (default) Start data recovery when there are enough seeders to ensure data loss does not occur. data_loss - Start data recovery immediately even if there is a chance of a data loss. fast_load_only - Start data recovery once all seeders are ready. robust_load_only - Start data recovery when there are enough seeders to ensure data loss will not occur. This option will ensure fast load is not attempted. force_load - Start data recovery immediately even if data loss will occur. Ignores all checks for required hosts/seeders and fast load is not attempted.
data	<p>Optional/Deprecated. Specifies whether the data from the space should be restored with the space. Specifying <code>with</code> indicates that the data from the space, which has been persisted to disk, should be restored with the space. Specifying <code>without</code> indicates the space should be restored without its data.</p> <p>The options <code>with data</code> and <code>without data</code> are now deprecated. Use <code>no_data_loss</code> and <code>no_data</code> instead respectively.</p>
quorum	Optional. An integer that specifies the minimum number of seeders which are needed for recovery to start. The default is 1.
keep_files	<p>Optional. A boolean value that indicates whether the persistence files should be saved or deleted after recovery is completed successfully. The default is <code>false</code>.</p> <p>If you do not specify the parameter when you issue the <code>recover space</code> command, all files used for recovery are deleted after a successful recovery. To clean up other files, use <code>cleanup datastore</code> command.</p> <p>If you specify <code>true</code> for the <code>keep_files</code> parameter, all files are retained when the space is recovered.</p>



When recovering a space which has a capacity defined, by default only the keys and indexes of the space are restored to memory regardless of whether or not the capacity has been reached. When capacity has been defined for a space, you can force the recovery of all data for a space, up to the capacity, by using the `robust_load_only` recovery policy. For any other recovery policy setting, the default behavior of only loading keys and indexes to memory is enforced.

Example

```
recover space 'myspace' no_data_loss
```



String values passed as parameters must be enclosed in either single or double quotes.

resume space

This command is used when shared-all persistence is specified for the space. Use this command on spaces that are in a `SUSPENDED` state to help them resume writing to the persister.

Syntax

```
resume space <string>
```

Parameters

The following table lists the parameters for this command with a description of each parameter.

resume space Parameters

Parameter	Description
space	Specifies the name of the space to resume. If shared-all persistence is set for the space and the space loses one of its persisters, then the space is set to the <code>SUSPENDED</code> state, which means that no writes to the persister can occur. You can issue the <code>resume space</code> command to put the space back in a normal state.

Example

```
resume space 'myspace'
```



String values passed as parameters must be enclosed in either single or double quotes.

cleanup datastore

This command provides you the ability to clean up the datastore for one or many spaces. With this command, all the old files and `INVALID` files get deleted.

In some scenarios, the old Shared Nothing files are not deleted, by default. Old Shared Nothing files are the ones that are not used for recovery or the ones that are not needed by a seeder that joins a space that is already in a `READY` state. These files never get deleted, not even when you use the `recover` command since this command only handles the files used for recovery and deletes any `INVALID` file involved in the recovery process. This is the where you need the `cleanup datastore` command. This command provides you the ability to clean up the datastore for one or many spaces. With this command, all the old files and `INVALID` files get deleted.

To delete such files on all the seeders or members use the following command:

```
execute on members cleanup datastore
```


To delete such files on a specific space, say "myspace", use the following command:

```
execute on members cleanup datastore "myspace"
```

Parameters

The following table lists the parameter for this command with a description of the parameter.

cleanup datastore Parameters

Parameter	Description
space	The name of the space to clean up. This is specified only if you want to clean up the invalid files on a particular space. The name of the space is specified within double quotes.

suspend persistence

This command suspends writes to shared-nothing files in case of disk or a file system failure such as, a longer or extended outage.

You can suspend persistence of a specific seeder or member and/or a specific space by providing the space name as the argument. By default, persistence is suspended for all spaces and members.

```
execute on members suspend persistence
```

Parameters

The following table lists the parameters for this command with a description of each parameter.

suspend persistence Parameters

Parameter	Description
space	Specifies the name of the space to suspend persistence.

resume persistence

This command resumes writes to shared-nothing files after a disk or a filesystem failure is fixed by the system administrator.

You can resume persistence of a specific seeder or member and/or a specific space by providing the space name as the argument. By default, persistence is resumed for all spaces and members.

```
execute on members resume persistence
```

Parameters

The following table lists the parameters for this command with a description of each parameter.

resume persistence Parameters

Parameter	Description
space	Specifies the name of the space to suspend persistence.

commit transaction

Commits an uncommitted transaction for a specified space member or for all metaspace members.

Syntax

```
commit transaction [all | member_name <string> | transaction_id <string>]
```

Parameters

The following table describes the parameters for this command.

commit transaction Parameters

Parameter	Description
all	Specify all to commit all transactions for all metaspace members.
member_name	Specify a member name to commit the uncommitted transactions for a specified space member.
transaction_id	Specify a transaction ID to commit a transaction with a specific transaction ID.



When you commit or rollback a transaction from **as-admin**, the member which began the transaction continues to remain in the transaction. So that member must either commit or rollback the transaction before it can start another transaction. However, the commit or rollback by the member will have no effect on the space since **as-admin** has already performed a commit or rollback on the original tuples in the transaction. To avoid this issue, use **execute on member** to cause the commit or rollback to be performed by the member owning the transaction. This command succeeds only if the member is still connected to the cluster.

rollback transaction

The **rollback transaction** command attempts to roll back the uncommitted transaction(s) for the metaspace member on which the command is invoked, provided the member is still connected to the cluster.

The all option causes all uncommitted transactions for all metaspace members to be rolled back. Specifying the name of a metaspace member causes the uncommitted transaction for the specified metaspace member to be rolled back. Specifying transaction_id will cause the transaction with the indicated id to be rolled back.

Syntax

```
rollback transaction [all | member_name <string> | transaction_id <string>]
```

Parameters

The following table describes the parameters for this command.

rollback transaction Parameters

Parameter	Description
all	Specify all to roll back all transactions for all metaspace members.
member_name	Specify a member name to roll back the committed transactions for a specified space member.
transaction_id	Specify a transaction ID to roll back a transaction with a specific transaction ID.



When you commit or rollback a transaction from `as-admin`, the member which began the transaction continues to remain in the transaction, if member is still connected to the cluster. So that member must either commit or rollback the transaction before it can start another transaction. However, the commit or rollback by the member will have no effect on the space since `as-admin` has already performed a commit or rollback on the original tuples in the transaction. To avoid this issue, use `execute on member` to cause the commit or rollback to be performed by the member owning the transaction. This command succeeds only if the member is still connected to the cluster.

set event_log

Use the `set event_log` command to enable or disable event logging.

Syntax

```
set event_log log_level <string> [log_file <string>]
```

Parameters

The following table describes the parameters for this command.

set event log Parameters

Parameter	Description
log_level	<p>A string that specifies the log level for the event logging. Can be one of the following:</p> <ul style="list-style-type: none"> • fatal • error • warning • info • fine • finer • finest • none <p>To disable event logging, specify <code>none</code>.</p>

Parameter	Description
log_file	<p>Optional. You can specify the log_file parameter to specify a file name or a fully qualified file name for the log file.</p> <p>Note that when specifying a fully qualified file name for the log file, Windows paths must have the backslash ('\') character escaped ('\\'). The default log file for logging events is:</p> <p><current directory>/log/as-events-<pid>.log</p> <p>where <i>pid</i> is the process ID of the as-admin process.</p>

set file_log

Use the `set file_log` command to enable or disable event logging.

Syntax

```
set file_log log_level <string>
    [log_file <string>]
    [log_limit <integer>]
    [log_append (true | false)]
    [log_count <integer>]
    [log_size <integer>]
```

Parameters

The following table describes the parameters for this command.

set file log Parameters

Parameter	Description
log_level	<p>A string that specifies the log level for the file logging. Can be one of the following:</p> <ul style="list-style-type: none"> • fatal • error, • warning • info • fine • finer • finest • none <p>To disable file logging, specify none .</p>

Parameter	Description
log_file	Optional. You can specify the log_file parameter to specify a file name or a fully qualified file name for the log file. Note that when specifying a fully qualified file name for the log file, Windows paths must have the backslash ('\') character escaped ('\\'). The default log file for logging events is: <code><current directory>/log/as-events-<i>pid</i>.log</code> where <i>pid</i> is the process ID of the as-admin process.
log_limit	Specifies an integer that sets the maximum size allowed for a log file, in bytes. The default value is -1 (no limit).
log_append	Specifies a boolean value that determines whether data can be appended to a log file. The default value is true.
log_count	Specifies an integer that determines the number of log files that the system retains. Use this option in combination with the log_size parameter to implement the rolling log file feature. With rolling log files, if a log file reaches the maximum size configured with the log_limit parameter, additional log files are created, up to the number of log files specified with the log_count parameter.
log_size	Specifies an integer that determines the maximum size of all log files combined.

show | describe console_log

Use the `show | describe console_log` command to indicate the current setting for console logging.

Syntax

```
show | describe console_log
```

Parameters

None.

show | describe member

Presents information and statistics about the specified member.

Syntax

```
(show | describe) member {<string> | [member_name <string>]  
[member_id <string>] [process_name <string>] [process_id <integer>]}  
[ stats [disk | nic] ]
```

Parameters

The table shows parameters for this command.

show | describe member Parameters

Parameter	Description
member	Optional. A String containing the member name or member ID of the metaspace member. If this option is not specified, at least one of the options described in the following rows must be specified.
member_name	Optional. Select members with specified member name.
member_id	Optional. Select members with the specified member id.
process_name	Optional. Select members whose process name matches the specified name.
process_id	Optional. Select members with the specified process ID.

Output

The following example shows output from the show member command.

```
show member 'test_member'
-----
Member id       : a62c8ce-c350-4febaa34-35e
Member name     : test_member
Member ip:port  : 10.98.200.206:50000
Member join time : 2012-06-28T00:49:56GMT
Member role     : manager
Member context  : none
-----
No spaces
```

Member Attributes**Member id**

The system-assigned member ID for the member.

Member name

If a name was specified when the member was created, shows the member name.

Member ip:port

Indicates the IP address and port number for the member.

Member join time

Indicates the time that the member joined the space.

Member role

Indicates the member role. Can be member or manager.

Member context

If the application that is managing the member has set up a thread context, displays the thread context; otherwise, indicates none.

show | describe members

Presents a short summary for each matching member connected to the metaspace including remote members.

Syntax

```
(show | describe) members [count] [<string> | [member_name <string>]
                                     [member_id <string>]
                                     [process_name <string>]
                                     [process_id <integer>]]
```

Remarks

In addition to the name and ID of each member of the metaspace, the output includes the *role* of the member in the metaspace — whether the member is a metaspace manager or just a metaspace member.

If remote members are active, the output shows the remote members.

Parameters

The following table describes the parameters for this command.

show | describe members Parameters

Parameter	Description
count	Optional. Shows the number of members connected to the metaspace instead of information about each member.
member	Optional. A String containing the member name or member ID of the metaspace member. If this option is not specified then at least one of the options described in the following rows must be specified.
member_name	Optional. Select members with specified member name.
member_id	Optional. Select members with the specified member id.
process_name	Optional. Select members whose process name matches the specified name.
process_id	Optional. Select members with the specified process ID.

show | describe metaspaces

The **show | describe metaspaces** command displays basic information about the metaspaces that a process is currently connected to.

For each metaspace that the process is connected to, the command shows:

- Metaspace Name

- Discovery URL
- Listen URL
- Member Name

For as-admin and as-agent, you can use this command to see which metaspace is currently connected, because as-admin and as-agent can only be connected to one metaspace at a time.

Syntax

```
show | describe metaspaces
```

Parameters

None.

show | describe space

Provides information about the space specified with the name parameter.

Syntax

```
show | describe space <string>
```

Remarks

See the Output section below for information on the output of the **show | describe space** command.

Parameters

show | describe space Parameters

Parameter	Description
space	The name of the space for which information is needed. The name value must be in single or double quotes.

Output

The output of the **show | describe space** command is displayed in the form of three tables as described in this section.

Space Definition Attributes

The first section of the output provides information about the space's attributes, arranged in the following columns:

Space Name

The name of the space, as given when the space was defined.

Space State

The current state of the space. The value can be *initial* (meaning that the Space needs more seeders or more persisters), *loading* (meaning the space is being loaded from the persistence layer), or *ready* (meaning the space has enough seeders, persisters, and has been loaded)

Distribution Policy

Can be either *distributed* (meaning the space is distributed) or *non_distributed*.

Replication count

The replication count is displayed. Default is 0, meaning there is no replication.

Replication Policy

Can be `sync` or `async`.

Capacity

The capacity value for the space in number of entries per seeder. -1 indicates that there is no capacity limit.

Eviction policy

The policy of the eviction to be applied when a space operation would cause the capacity to be exceeded. Can be `none` (no eviction) or `LRU` (least recently used eviction).

Min seeders

The minimum number of seeders that need to be joined to the space before the space becomes ready.

Persistence Type

Can be `none`, `share_all`, or, `share_nothing`.

Persistence Policy

Can be `none`, `sync` or `async`.

Update transport

The transport protocol used to distribute notifications of updates to the data stored in the space. Can be `unicast` or `multicast`.

Entry TTL

The TTL (time-to-live) of the entries stored in the spaces in milliseconds. Default is -1 (forever).

Lock wait

The **Lock wait** defined for the space is displayed. Default is -1 (forever).

Lock TTL

The **Lock TTL (time-to-live)** defined for the space is displayed. Default is -1 (forever).

Lock scope

Indicates the lock scope for the space, if set. Can be `thread` or `process`.

Space wait

Indicates the space wait value for the space, if a space wait value has been set.

Write timeout

Indicates the write timeout value for the space, if a write timeout has been set.

Read timeout

Indicates the read timeout value for the space, if a read timeout has been set.

Version num

Indicates how many time the space definition has been changed since it was defined initially. This value is incremented for each alter space operation. The initial value is 0.

Fields

For each field, the name and type of the field, and whether or not the field is nullable (optional), is displayed.

Below this list of field definitions are names of all the fields that make up the space's key.

Key and Index

The display shows any primary keys that have defined for the space and secondary indexes. For each key or index, the output indicates the index type (HASH or TREE) and the fields for that key or index definition.

Key : {KeyDef index_type=HASH, fields=[KEY, field0, field1, field2]}

In addition, the display indicates any distribution fields that have been set up:

Distribution Fields: {[KEY, field0, field1, field2] }

Distribution fields are key fields that have been configured for affinity—for these fields, all tuples that have the same value are stored on the same seeder.

Members

List the members of the space.

Statistics

The count of put, get, take, seeded, and replicated entries is displayed, as well the number of expires, evictions, locks, unlocks and invoke, as well as a ToPersist count, which indicates how many tuples are required to be persisted to the database if the write-behind feature is configured. These counts are displayed per member and the cumulative count for all members connected to this space.

show | describe space entries

Lists the entries in a specified space.

Syntax

```
show | describe space <string>
entries [filter <string>]
[prefetch <integer>]
[query_limit <integer>]
[distribution_scope ('all' | 'seeded')]
[time_scope ('current' | 'snapshot')]
[output ('console' | <file name string>)]
```

Parameters

The following table describes the parameters for this command.

show | describe space entries Parameters

Parameter	Description
filter	Specifies a filter value that limits the output to entries matching the filter criteria.
prefetch	Specifies the number of entries to fetch from the space each time the browser retrieves more entries to buffer. The default prefetch value for snapshot browsers is 0, while the default prefetch value for current browsers is 1.
query_limit	Indicates the maximum number of entries a query on this space can return (default 10000).

Parameter	Description
distribution_scope	Indicates whether all space entries should be browsed or just those which are seeded on this member. The default is to browse all entries.
time_scope	Specifies to indicate which type of browser to use. The default time_scope is current.
output	<p>Specifies whether the results of the command are saved in a file or displayed on the console. By default, the results of any as-admin command are returned to the as-admin member and displayed on its console.</p> <p>When used with the execute on command to route the command, specifying output 'console' causes the results of the show space entries command to be displayed on the executing member's console.</p> <p>Any string other than 'console' for the output option is assumed to be a filename. If a filename is specified, the results of the show space entries command are written to the indicated file. If the file already exists, the results are appended to the existing file contents.</p>



The `-entry_scope` parameter of the `show | describe space <string> entries [filter <string>]` command is deprecated. Instead, use the `-distribution_scope` parameter.

show | describe space locks

Provides information about each entry in a space that is locked, including which member of the space has locked the entry and the entry's field values.

Syntax

```
show | describe space <string>
locks [count]
[member_name <string>] [filter <string>]
```

Parameters

The following table describes the parameters for this command.

show | describe space locks Parameters

Parameter	Description
count	Specifies display of the number of locks.
member_name	<p>Specifies display of only the locks for a specified member. For example, to see a count of the locks for a particular space member you could enter the command:</p> <pre>show space 'myspace' locks member_name 'member1' count</pre>

Parameter	Description
filter	<p>Specifies display of only the locks that match a specified filter string.</p> <p>For example, to see if there is an existing lock on a particular space entry you could enter the command:</p> <pre>show space 'myspace' locks filter 'key=5'</pre>

show | describe space size

Lists the number of seeded entries in a space.

Syntax

```
show | describe space <string>
size [filter <string>]
[map] [all | replication_degree <integer>]
```

Parameters

The following table describes the parameters for this command.

show | describe space size Parameters

Parameter	Description
filter	Specifies a filter value that limits the display to how the number of entries which satisfy the filter string.
map	Specifies display of a map that breaks down the number of entries by space member.
all	Specifies that the display includes replicated entries.
replication_degree	Specifies an integer that limits display to entries of a specified replication degree.

show | describe space query

Provides information on cost of execution of each seeder. The cost of a query execution can be measured by the time taken to execute a query. If there are indexes, the cost of execution is measured by the number of entries. If not, it is measured by the time taken to execute the query.

Syntax

```
show | describe space <string> query [filter <string>]
```

Parameters

The following table describes the parameters for this command.

show | describe space query Parameters

Parameter	Description
space	The name of the space for which information is needed. The name value must be in single or double quotes.
filter	Specifies a string value that limits the execution to entries matching the filter criteria.

show | describe query

Displays information about queries in the metaspace.

Syntax

```
(show | describe) query [detail] [state <string>]
                                [space <string>]
                                [member_id <string>]
                                [member_name <string>]
```

Parameters

The following table describes the parameters for this command.

show | describe space query Parameters

Parameter	Description
detail	By default, the information displayed in the output is in a tabular format. Specifying this option would switch the output to a detailed format.
state	Specifies the state of the query. When not specified, all queries are displayed. Can be one of the following states: <ul style="list-style-type: none"> • running - Shows only running queries • completed - Shows only completed queries
member_id	Shows query from the member with the specified member_id.
member_name	Shows query from the member with the specified member_name.

show | describe spaces

Lists all spaces defined for the metaspace.

Syntax

```
show | describe spaces [all]
```

Parameters

The following table describes the parameters for this command.

show | describe spaces Parameters

Parameter	Description
all	Specifies display about information regarding system spaces, in addition to information about user-defined spaces.

show | describe transactions

Lists all spaces defined for the metaspace.

Syntax

```
show | describe transactions
[member_name <string>|
state ('uncommitted' | 'committed' | 'rolled_back')]
```

Parameters

The following table describes the parameters for this command.

show | describe transactions Parameters

Parameter	Description
member_name	Specify a member name for which to show transactions.
state	Specifies a transaction state for which to show transactions. Can be one of the following: <ul style="list-style-type: none"> • uncommitted Show uncommitted transactions. • committed Show committed transactions. • rolled_back Show rolled back transactions.
transaction_id	Specify a transaction ID to roll back a transaction with a specific transaction ID.

show | describe event_log

Use the `show | describe event_log` command to indicate the current setting for event logging.

Syntax

```
show | describe event_log
```

Parameters

None.

show | describe file_log

Use the `show | describe file_log` command to indicate the current setting for file logging.

Syntax

```
show | describe file_log
```

Parameters

None.

show | describe member stats

If performance monitoring is enabled, displays system performance statistics for the specified member.

Syntax

```
show | describe member <string> stats
```

show | describe member stats Parameters

Parameter	Description
member	The member parameter specifies the name of the space member for which to display performance statistics.
stats	The as-admin <code>show describe member <string></code> command displays the statistic mentioned in the section, Statistics Attributes

Statistics Attributes

The as-admin `show | describe member <string>` command displays the following information:

Member Name

Indicates the member name or the agent.

Timestamp

Indicates the timestamp for the data that is displayed.

AS Version

Indicates the ActiveSpaces version that is running.

User Name

Indicates the user name of the user running the process.

Command Name

Indicates the command used to start as-admin; for example, `as-admin -monitor_system true`.

Process ID

Indicates the process ID for the process that is monitored.

System Name

Indicates the operating system that is running.

CPU Count

Indicates the number of CPUs running on the system.

Resident Memory (MB)

Indicates the amount of physical memory currently allocated to the member or agent process.

Page File Size (MB)

Indicates the current size of the system pagefiles allocated by the system.

JVM Finalizing Count

The amount of memory freed by the finalize operation on the JVM.

CPU Load

Indicates the load on the CPU, in MB.

Mem Load

Indicates the memory load on the system.

Threads

Indicates the number of threads running for the process.

JVM Heap

Indicates the following information for the JVM heap:

- Peak (MB) - Indicates the peak usage in MB.
- Committed MB - The committed heap usage.
- Max MB - The maximum heap usage.

JVM Nonheap

Indicates the following information for the JVM:

- Peak (MB) - Indicates the peak nonheap usage in MB.
- Committed MB - The committed nonheap usage.
- Max MB - The maximum nonheap usage.

show | describe system stats

If performance monitoring is enabled, displays system performance statistics for all of the members of the space.

Syntax

```
show | describe system stats
```

Parameters

None.

Statistics Attributes

The as-admin **show | describe member system stats** command displays the following information:

Member Name

Indicates the member name or the agent.

Timestamp

Indicates the timestamp for the data that is displayed.

AS Version

Indicates the ActiveSpaces version that is running.

User Name

Indicates the user name of the user running the process.

Command Name

Indicates the command used to start as-admin; for example, `as-admin -monitor_system true`.

Process ID

Indicates the process ID for the process that is monitored.

System Name

Indicates the operating system that is running.

CPU Count

Indicates the number of CPUs running on the system.

Resident Memory (MB)

Indicates the amount of physical memory currently allocated to the member or agent process.

Page File Size (MB)

Indicates the current size of the system pagefiles allocated by the system.

JVM Finalizing Count

The amount of memory freed by the finalize operation on the JVM.

CPU Load

Indicates the load on the CPU, in MB.

Mem Load

Indicates the memory load on the system.

Threads

Indicates the number of threads running for the process.

JVM Heap

Indicates the following information for the JVM heap:

- Peak (MB) - Indicates the peak usage in MB.
- Committed MB - The committed heap usage.
- Max MB - The maximum heap usage.

JVM Nonheap

Indicates the following information for the JVM:

- Peak (MB) - Indicates the peak nonheap usage in MB.
- Committed MB - The committed nonheap usage.
- Max MB - The maximum nonheap usage.

abort query

Allows for canceling running queries. On aborting the query, the application gets a `REQUEST_CANCELED` error message.

Syntax

```
abort query [member_id <string>]
           [member_name <string>]
           [space <string>]
```

Parameters

The following table describes the parameters for this command.

show | describe space query Parameters

Parameter	Description
member_id	Abort queries running on the specified member.
member_name	Abort queries running on the specified member.
space	Abort queries on the specified space

unlock

The `unlock` command is used to force the unlocking of entries in a space. Entries can be unlocked using either the `lock_id` parameter or the `all` parameter. The `lock ID` of a space entry can be discovered using the `show space <space_name> locks` command.

You can use the command parameters to:

- Unlock all entries in a space (For example, `unlock space 'myspace' all`)
- Unlock all entries locked by a particular space member (For example, `unlock space 'myspace' all member_name 'member1'`)
- Unlock entries that satisfy a filter string (For example, `unlock space 'myspace' all filter 'state=CA'`)
- Unlock a subset of entries locked by a particular space member (For example, `unlock space 'myspace' all member_name 'member1' filter 'total>1000'`)
- Unlock a space entry that has a particular key value (For example, `unlock space 'myspace' all filter 'key=5'`).

Syntax

```
unlock space <string>
(lock_id <string> | all
 [member_name <string>] [filter <string>])
```

Parameters

The following table describes the parameters for this command.

unlock space Parameters

Parameter	Description
space	Specifies the name of the space to unlock.
lock_id	Specifies the lock ID for the space to be unlocked.
all member_name <string>	Unlock all entries locked by a specified member. For example, you can specify: unlock space 'myspace' all member_name 'member1'
filter	A string that specifies a filter to select entries to unlock.

validate policy_file

Use the **validate policy_file** command to validate the syntax and settings in a specified security policy file.

Syntax

```
validate policy_file <string>
    [policy_name <string>]
    [domain_identity_password <string>]
```

Parameters

The following table lists the parameters for this command with a description of each parameter.

validate policy_file Parameters

Parameter	Description
policy_file	Specifies the name of the security policy file to be validated.
policy_name	Optional. Specifies the name of a specific security policy to be validated.
domain_identity_password	Optional. The password for the domain identity key in the security policy file. If not specified and the key requires a password, you are prompted to supply the password.

Example

The following examples illustrate the syntax of the **validate policy_file** command:

- `validate policy_file 'policy.txt'`
- `validate policy_name 'mypolicy' policy_file 'policy.txt'`



String values passed as parameters must be enclosed in either single or double quotes.

validate token_file

Use the `validate token_file` command to validate a security token file that you have generated by using the `define | create token_file` command.

Syntax

```
validate token_file <string>
```

Parameters

The following table lists the parameters for this command with a description of each parameter.

validate token_file Parameters

Parameter	Description
token_file	Specifies the name of the token file to be validated.
token_identity_password	Optional. The password for the identity key in the security token file. If not specified and the key requires a password, you are prompted to supply the password.

Example

The following examples illustrate the syntax of the `validate token_file` command:

- `validate token_file "mytoken"`



String values passed as parameters must be enclosed in either single or double quotes.

validate truststore

Use the `validate truststore` command to validate a specified security certificate file.

ActiveSpaces validates the specified truststore and notifies with error messages, if errors are found. If the certificate file is validated successfully, the output indicates the X509 information and validity time period.

Syntax

```
validate truststore cert_file <string>
```

Parameters

The following table lists the parameters for this command with a description of each parameter.

validate truststore Parameters

Parameter	Description
cert_file	Specifies the file path name of the certificate file that is to be validated.

Example

The following examples illustrate the syntax of the `validate truststore` command:

```
validate truststore cert_file "mycert.pem"
```



Parameter values must be enclosed in either single or double quotes.

The following example shows sample output from the command:

```
as-admin> validate truststore cert_file "mycert.pem"
[1] /CN=AS-DOMAIN-FEF3A467 [serial: A9CD21B4A05FB244]
[valid after: Jan 17 03:10:34 2013 GMT, before: Jan 17 03:10:34 2014 GMT]
Found [1] requestor trust anchor(s)
Truststore validated successfully: mycert.pem
```

as-admin Commands for Cross-site Replication

The **as-admin** utility offers some commands for cross-site replication.

To see a list of all **as-admin** commands related to cross-site replication, use the following command:

```
help site
```

Enter **help -long** before the name of a specific **as-admin** command to display detailed information about that command. For example, the following command lists the syntax, command description, and parameter descriptions of the **alter site** command.

```
help -long alter site
```

alter site

The **alter site** command is used to change a characteristic of the specified remote site. When you alter a remote site, the site is first deleted and then created again with the new changes. This helps in re-establishing connections to the remote site. This is especially useful when the router discovery URL has changes.

Syntax

```
alter site <string>
  ** specify at least one of the following **
  router_discovery <string>
  suspend <boolean>
```

Parameters

The parameters for this command are listed in the following table:

alter site Parameters

Parameter	Description
site_name	Name of the site to alter.
router_discovery	A string containing a new router discovery URL to use for connecting to the routers of the remote site.
suspend	Toggles whether routing to the remote site starts immediately or not. The default is <code>false</code> indicating routing starts immediately when the remote site is re-created.

A remote site must first be suspended before **alter site** is called to modify the site definition.

create site

The **create site** command is used to define a remote site. A remote site must be defined before operations or transactions can be routed to it.

Syntax

```
create | define site <string> [router_discovery <string>] [suspend <boolean>]
```

Parameters

The parameters for this command are listed in the following table:

Property	Description
site_name	The name of the remote site.
router_discovery	A URL containing the listen URLs of each as-router on the remote site.
suspend	Toggles whether routing to the remote site is suspended or started immediately. The default is to start routing to the remote site immediately.

drop site

The **drop site** command removes the specified remote site definition from the metaspace.

Syntax

```
drop site <string>
```



The remote site must be suspended prior to invoking the **drop site** command.

Parameters

drop site Parameters

Parameter	Description
site_name	Name of the site to remove.

resume site

The **resume site** command resumes routing to the site which was previously suspended.

Syntax

```
resume site <string>
```

Parameters

resume site Parameters

Parameter	Description
site_name	The name of the remote site which was previously suspended and should be resumed.

show local site

The `show local site` command displays the settings of the local site.

Syntax

You can use any of the following commands to display the local site:

```
show site
show local site
show site <string>
```

Parameters

show local site Parameters

Parameter	Description
site_name	The name of the local site.

show remote site

The `show remote site` command displays the settings of the remote site.

Syntax

You can use any of the following commands to display the remote site:

```
show remote site
show site <string>
show remote site <string>
```

Parameters

show remote site Parameters

Parameter	Description
site_name	The name of the remote site.

show sites

The `show sites` command displays detailed information about both, local site and remote site.

Syntax

```
show sites
```

Parameters

None.

suspend site

The `suspend site` command suspends routing to the specified remote site.

Syntax

```
suspend site <string>
```

Parameters

suspend site Parameters

Parameter	Description
site_name	The name of the remote site to suspend.

site prompt

The `site prompt` command applies only to the `as-admin` executable. The command toggles the display of the name of the local site as part of the `as-admin` command prompt.

Syntax

By default, the name of the local site is included as a part of the `as-admin` command prompt, which means that the default is `on`.

```
site prompt on | off
```

For example, when it is `on`, the prompt format is as follows:

```
as-admin.primarySite>
```

Parameters

site prompt Parameters

Parameter	Description
on	Include the name of the local site as part of the <code>as-admin</code> command prompt.
off	Display the normal <code>as-admin</code> command prompt.

The as-dump Utility

You can use the as-dump utility to view backup files that are used with ActiveSpaces shared-nothing persistence.

The as-dump program is a utility that you can run offline to examine shared-nothing persistence files and get information such as the ActiveSpaces version the file was created with, the number of entries in the file, and optionally, the data in the file entries. As-dump is useful for examining the data store files created when you are using shared-nothing persistence and detecting possible problems.

The as-dump utility takes a file name and/or a specific directory name as input. You must provide the full directory path.

If you provide the directory name, as-dump reads all files in the directory or in its subdirectory and prints out information on each of the files. If you specify a filename, as-dump outputs information for the specified file only.

Syntax

The as-dump utility has the following syntax:

```
as-dump [-vhm] <filepath/directory>
-h help
-v verbose, verbose output
-m <int>, number of entries to print at once
```

as-dump Parameters

Parameter	Description
<i>filepath/directory</i>	Required.
-h	Displays the command syntax for as-dump.
-v	Verbose. Displays detail for each tuple in the file.
-m <int>	Specifies output of <i>int</i> entries.

Sample Output

The output from as-dump indicates:

- The location of the persistence file (or files if you specify the directory name only).
- The filename for each persistence file.
- The ActiveSpaces version that created the files.
- The number of entries in the file.
- If you specify the verbose option, the contents of each tuple.

The following example shows output from the as-dump command with the verbose option and a specified filename specified:

```
as-dump -v c:\tmp\ms\shared_nothing_persisted\
a62c937-c350\a62c937-c350_store_1352146870
[2012-11-05 14:29:23:101][4896][7040][INFO][asdump]
fileloc=c:\tmp\ms\shared_nothing_persisted\a62c937-c350\a62c937-
c350_store_1352146870
[2012-11-05 14:29:23:101][4896][7040][INFO][asdump]
<?xml version="1.0" encoding="UTF-8"?>
[2012-11-05 14:29:23:201][4896][7040][INFO][asdump]
<Dump>
```

```

[2012-11-05 14:29:23:301][4896][7040][INFO][asdump]
<Filter></Filter>
[2012-11-05 14:29:23:401][4896][7040][INFO][asdump]
<File>
[2012-11-05 14:29:23:501][4896][7040][INFO][asdump]
<FilePath>c:\tmp\ms\shared_nothing_persisted\a62c937-c350\a62c937-
c350_store_1352146870</FilePath>
[2012-11-05 14:29:23:601][4896][7040][INFO][asdump]
<Version>2.0.2</Version>
[2012-11-05 14:29:23:701][4896][7040][INFO][asdump]
<ViewNumber>2</ViewNumber>
[2012-11-05 14:29:23:801][4896][7040][INFO][asdump]
<Entries>
[2012-11-05 14:29:23:901][4896][7040][INFO][asdump]
<Tuple>
  <value type="int32" pos="0" name="key"> 1 </value>
  <value type="string" pos="1" name="value"> one </value>
  <value type="date_time" pos="2" name="time"> 2012-11-05T20:21:33.000GMT </
value>
</Tuple>
[2012-11-05 14:29:24:1][4896][7040][INFO][asdump]
<Tuple>
  <value type="int32" pos="0" name="key"> 2 </value>
  <value type="string" pos="1" name="value"> two </value>
  <value type="date_time" pos="2" name="time"> 2012-11-05T20:21:41.000GMT </
value>
</Tuple>
[2012-11-05 14:29:24:101][4896][7040][INFO][asdump]
<Tuple>
  <value type="int32" pos="0" name="key"> 3 </value>
  <value type="string" pos="1" name="value"> three </value>
  <value type="date_time" pos="2" name="time"> 2012-11-05T20:21:54.000GMT </
value>
</Tuple>
[2012-11-05 14:29:24:201][4896][7040][INFO][asdump]
<Tuple>
  <value type="int32" pos="0" name="key"> 4 </value>
  <value type="string" pos="1" name="value"> four </value>
  <value type="date_time" pos="2" name="time"> 2012-11-05T20:22:02.000GMT </
value>
</Tuple>
[2012-11-05 14:29:24:301][4896][7040][INFO][asdump]
<Tuple>
  <value type="int32" pos="0" name="key"> 5 </value>
  <value type="string" pos="1" name="value"> five </value>
  <value type="date_time" pos="2" name="time"> 2012-11-05T20:22:11.000GMT </
value>
</Tuple>
[2012-11-05 14:29:24:424][4896][7040][INFO][asdump]
</Entries>
[2012-11-05 14:29:24:524][4896][7040][INFO][asdump]
  <EntryCount>5</EntryCount>
[2012-11-05 14:29:24:624][4896][7040][INFO][asdump]
  <InvalidBlockCount>0</InvalidBlockCount>
[2012-11-05 14:29:24:724][4896][7040][INFO][asdump]
  <ExceptionCount>0</ExceptionCount>
[2012-11-05 14:29:24:824][4896][7040][INFO][asdump]
</File>
[2012-11-05 14:29:24:924][4896][7040][INFO][asdump]
</Dump>

```

The as-agent Utility

ActiveSpaces provides an agent process called as-agent. The main purpose of as-agent is to join all distributed spaces in the specified metaspace as a seeder. You can run as-agent on any host.

You can also use as-agent to:

- Ensure that the desired degree of replication specified for a space is achieved.

Because the amount of data that can be stored in a space depends on the number of seeding members of the space, you might need to add seeders to a space to scale it up. Although any process using the ActiveSpaces API can become a seeder on a space, there are situations where applications may not always be running and joined to the space, or where not enough applications have joined the space.

- Ensure data integrity.

Because the data contained in the space disappears along with the last seeder in the space, this can be a problem. You can use the as-agent process to ensure that there are always one or more seeders for each space in the metaspace.

Agents remain connected to the metaspace, and always seed the system spaces, keeping the metaspace's space definitions alive, even when all other applications have quit the metaspace.

Running as-agent to Include a Command Console

You can use as-agent to run Admin CLI commands when as-agent starts up. To run as-agent and display a command prompt which accepts Admin CLI commands, specify the `-admin` parameter as follows:

```
java -jar as-agent.jar -admin
```

When as-agent starts, an as-agent command prompt appears, where you can enter the Admin CLI commands.

Remote Invocation

The use of remote invocation is possible when using the Java as-agent. Use the following command to start your Java as-agent so that it can find the class files it should use for remote invocation:

```
java -Djava.ext.dirs=$LOCATION_OF_JARS$ -jar as-agent.jar
```

Otherwise, running java as-agent in a cluster that includes remote members will cause invoke methods to fail with a "ClassNotFoundException" exception.

Starting as-agent

as-agent is provided for Java, .NET, and C.

Procedure

1. Ensure the ActiveSpaces `/bin` directory is in defined in the `path` variable. Go to the ActiveSpaces `\bin` directory:
 - **Windows:**
Default location: `<TIBCO_HOME>\as\<version>\bin`
 - **UNIX/Linux:**
Default location: `/opt/tibco/as/<version>/bin`

2. For Java, ensure as-agent.jar from the ActiveSpaces /lib directory is in your CLASSPATH.
3. Start as-agent as follows

Platform	Command
Native executable	as-agent
.NET executable	Agent.NET
Java	java -jar as-agent.jar

You can start as-agent with any of the following parameters:

```
-metaspace <metaspace_name>
-discovery <discovery_url>
-listen <listen_url>
-member_name <member_name>
-remote_listen <remote_listen_url>
-log <file_path>
-log_debug <log_level>
-loglimit <logfile_size_limit>
-logcount <number_of_log_files>
-logappend <true | false>
-debug <log_level>
-advisory_level <advisory_level>
-name <member_name>
-admin
-data_store <directory path>
-worker_thread_count <count>
-rx_buffer_size <size>
-security_policy <string>
-security_token <string>
-authentication_domain <string>
-authentication_username <string>
-authentication_keyfile <string>
-authentication_password <string>
-identity_password <string>
-monitor_system true|false
-input <filepath>
-member_timeout <ms>
-cluster_suspend_threshold
-connect_timeout <ms>
-autojoin.role <role>
```

Each parameter specifies a parameter name and a value.



If you start as-agent with the `-member_name` parameter; for example `as-agent -member_name "agent1"`, then when you run the Admin CLI you can issue the `show member` command and specify the member name of the agent to display the attributes of the member; for example, `show member_name "agent1"`

To display usage help for as-agent, enter `as-agent -help`

The output of the help command includes the default values used if no parameter is provided.



When you are using the `-data_store` option with the `as-agent` command, such as `as-agent -datastore "c:/abc/tmp"` ensure that you specify the absolute path to a directory. Also ensure that the datastore has write permissions on the directory.

as-agent Command Parameters


The command line parameters for as-agent allow you to specify configuration parameters that control how the agent functions.

The following table describes the command parameters for as-agent:

as-agent Command Parameters

Parameter	Description
-metaspace <metaspace_name>	Specifies the name of the metaspace that as-agent is to join.
-discovery <url> / <url_list>	<p>Specifies the discovery URL to be used to discover the space. The discovery URL format can have the following formats:</p> <ul style="list-style-type: none"> • tcp://interface:port;interface2:port2;interface3:port3—Specifies TCP discovery. • tibpgm://dport/interface;multicast/key1=value1;key2=value2;key3=value3—specifies TIBCO PGM discovery. This is the default value <p>For more information on the -discovery parameter, see the documentation for the discovery parameter that is used with the with the as-admin connect command (see connect).</p>
-listen <url>	Specifies a listen URL for the as-agent. The default value is TCP. For more information, see the description of the listen parameter that is used with the with the as-admin connect command (see connect).
-remote_listen <url>	<p>Specifies a remote listen URL that is used to contact a remote client and accept incoming remote client connections. The URL format is:</p> <p>tcp://interface:remote_listen_port</p>
-log	<p>(optional) Specifies the path to the log file and the log file name.</p> <p>If you include the parameter <code>-log <log_file></code>, then the log filename will be <code><log_file>-<processid>.log</code>. For example, if you enter <code>-log as</code>, then the log filename is <code>as-<processid>.log</code>.</p>
-log_debug	<p>Specifies a debug log level that controls the level of log messages written to the log file, To specify a log level, enter:</p> <p><code>-log_debug <LogLevel></code></p> <p>The log levels are as follows:</p> <p>1 = ERROR_LEVEL</p> <p>2 = WARNING_LEVEL</p> <p>3 = INFO_LEVEL</p> <p>The default is 3 (INFO_LEVEL). The log information is written to the log. If a log file is not specified, then the debug (log level) value is ignored.</p>
-log_limit	Specifies the maximum log file size in bytes before rollover. If log file rollover is configured, a new log file is started when the log file limit is reached.

Parameter	Description
-log_count	Specifies the number of log files that can be created. the default is one log file. If rolling logs are enabled, specifies the maximum number of log files.
-log_append	Specifies whether data can be appended to the log file. The default value is true.
-debug <log_level>	Specifies a debug log level that controls which level of log messages are written to console. For more details, see - log_debug.
-advisory_level <advisory_level>	Not implemented in the current release.
-member_name <membername>	Specifies a member name for the member. This helps to identify which member name is associated with which member ID. The show members command displays the member name if one has been assigned; otherwise, a default member name is assigned that is constructed from the member ID.
-data_store <directory path>	If you are using shared-nothing persistence, specifies the directory where persistence data is stored.
-worker_thread_count <count>	Specifies number of threads that can be used for program invocation. Default is 32, meaning you can have 32 invocations in parallel.
-rx_buffer_size <size>	Specifies the TCP buffer size for receiving data. The default value is 2 MB. If an application is using large tuples, this value can be increased accordingly. If the value is smaller than what is needed to receive over the connection, then a dynamic buffer is allocated during the lifetime of the message. It is better to use a large value to avoid a context switch from a dynamic to a static buffer
-security_policy <string>	If TIBCO ActiveSpaces security is implemented and you are connecting from a domain security controller node, specify the security_policy parameter and provide the directory path and filename for the policy file. If you specify a policy file, do not specify the security_token parameter.

Parameter	Description
-security_token <string>	<p>Specifies the token file for a security domain requestor that must be authenticated by a security domain controller. If TIBCO ActiveSpaces security is implemented and you are connecting from a requestor node, and the metaspace to which you are connecting requires a token file, specify the security_token parameter and provide the directory path and filename for the token file. If you specify a token file, do not specify the security_policy parameter.</p> <p>The following example shows how to start as-agent as a requestor node with security enabled:</p> <pre>java -jar as-agent.jar -metaspace ms -security_token "exdomain_token.txt"</pre> <p> Remember that LDAP authentication is supported only with Java agents or the Java API.</p>
-authentication_domain <string>	The name of the windows domain to log into. If local/ntlm account (as per the controller), "." can be used. If not windows, it will be ignored
-authentication_username <string>	The authentication username of the user account.
-authentication_keyfile <string>	The pkcs12 keyfile location of the user to be logged in as (sasl/external x509 ldap auth)
-authentication_password <string>	The authentication password of the user account to be logged in as or the password of the pkcs12 key file if x509 auth is used.
-identity_password <string>	The policy's security domain's or the token's identity password if the identity is encrypted
-monitor_system true false OR <boolean>	<p>The -monitor_system parameter allows you to enable the system monitor on the specified member to start and advertise its performance statistics in its joined metaspace(s) scope.</p> <p>To start performance monitoring with as-agent or as-admin, start the utility with the -monitor_system parameter set to true, for example:</p> <pre>as-agent -monitor_system true</pre> <p>The default value is false.</p>
-input <filepath>	The -input parameter allows you to pass a file containing Admin CLI commands as input to as-agent
-admin	Runs the as-agent utility and displays a command window which accepts Admin CLI commands.
-member_timeout	Specifies the time in milliseconds to wait for a member to reconnect. The default is 30000.
-cluster_suspend_threshold	Specifies the lost hosts allowed before membership operations are suspended. By default, it is not suspended.

Parameter	Description
-connect_timeout	Specifies the time to wait to connect to the metaspace.
-autojoin.role	When as-agent is run, it first connects to the metaspace and then joins any new or existing spaces in the metaspace as a seeder for each space. The <code>-autojoin.role</code> option allows you to override that behavior and have the as-agent always join spaces as a leech. For example, <code>as-agent -autojoin.role leech</code> . This setting is typically used when starting an as-agent that will only act as a proxy for remote clients.



The member name that you specify with the `-member_name` parameter can be the member name; or, if you are implementing host-aware replication, can specify a member name in the form *a.b*, where *a* specifies the name of a region, for example *region1*, and *b* specifies the name of a seeder running in that region, in effect, on the same host.

For information on deploying host-aware replication, see [Host-Aware Replication](#).

For more details about discovery and listen URLs, see *TIBCO ActiveSpaces Developer's Guide*.

The as-convert Utility

The as-convert utility converts ActiveSpaces shared-nothing files from one format (or one version) to another (usually higher).

The utility does the following:

- If a file name is provided, processes that file and prints out the result.
- If a space name is provided along with the metaspace name, converts all files that belong to that space.
- If a member name is provided along with the space name and metaspace name, converts files for that member only.
- If a metaspace name is provided, the utility does the above for each space that is part of the metaspace.
- If no argument is provided, processes the entire data store — reads each subdirectory and converts all files.

The `dry_run` option will just touch the files and identify the files that are older than the current version. This option is good for estimating how many files need conversion.

If you are upgrading from release 2.0.x to release 2.1.x or higher, then you must run the as-convert utility to upgrade shared-nothing persistence files to the format for newer releases. The `as-convert.exe` file is located in the following directory: `AS_HOME/bin`



Before you run `as-convert`, stop all as-agents and seeders. The `as-convert` utility must be run offline.

Command Syntax

```
as-convert -data_store <directory_path> -metaspace <metaspace_name> -space
<space_name>
-member_name <membername> -file <file_name> -compact -dry_run -verbose -log
<log_file> -debug <log_level>
```

The following table describes the parameters for `as-convert`.

Parameters for as-convert

Parameter	Usage
<code>-data_store</code>	<code>directory_path</code> specifies the path to the data store to convert.
<code>-metaspace</code>	To specify conversion of the data files for all of the spaces defined for a metaspace, specify a metaspace name.
<code>-space</code>	To specify conversion of the data files for a specific space within the metaspace, specify the space name with the <code>-space</code> parameter and the metaspace name with the <code>-metaspace</code> parameter.
<code>-member_name</code>	To specify conversion of the data files for a specific space member, specify the member name with the <code>-member_name</code> parameter, the space name with the <code>-space</code> parameter, and the metaspace name with the <code>-metaspace</code> parameter.

Parameter	Usage
-file	To specify conversion of a specific file, specify the filename with the <code>-file</code> parameter, the data store path with the <code>-data_store</code> parameter, and the metaspace name with the <code>-metaspace</code> parameter.
-compact	Deletes any white spaces in converted file. White spaces can be added to the data file as a result of processing Takes, which delete data.
-dry_run	To run the utility without actually converting the data store and output informational messages, include the <code>-dry_run</code> parameter.
-help	Outputs a summary of the command syntax for <code>as-convert</code> .
-verbose	Causes output of more information.
-log	Specifies the name of a log file to which to write log information
-debug	Specifies the log level for messages output by the utility.

Administering ActiveSpaces Security

ActiveSpaces provides the `as-admin` utility to configure and administer the security aspect of ActiveSpaces. You can also use the ActiveSpaces API to manage access to secured metaspaces.

Basic Entities Involved in Security

Configuring and maintaining security involves the following elements:

as-admin utility

Sets discovery parameters, generates and maintains security configuration files.

policy files

Specifies security settings across metaspaces, binds metaspaces to security domains.

token files

Define connection parameters to secured metaspaces.

ActiveSpaces API

Sets up and manages access to secured metaspaces.

Main Tasks for Setting Up Security

[Table 36, Tasks for Setting Up Security](#) lists the main tasks for setting up ActiveSpaces security.

Tasks for Setting Up Security

Task	See
Create a Policy File	Creating a Security Policy File
Edit the Policy file	Edit a Security Policy File
Set up Data Encryption	<p>TIBCO ActiveSpaces allows you to specify encryption of tuple data for fields that have been defined as secure data fields.</p> <p>Data encryption is set up in the policy file for each domain and by using the TIBCO ActiveSpaces security API functions.</p> <p>For detailed information on implementing data encryption, see <i>TIBCO ActiveSpaces Developer's Guide</i>.</p>
Validate the Security Policy file	Validating a Security Policy File
Create a Security Token	Creating a Security Token
Validate a Security Token	Validating a Security Token File

Task	See
Set up Authorization	<p>If you want to provide granular authorization, ActiveSpaces allows you to use using Access Control Lists (ACLs) to set up authorization scopes, rights, and privileges.</p> <p>For information on setting up authorization, see <i>TIBCO ActiveSpaces Developer's Guide</i>.</p>
Start Security Domain Controllers	Starting Security Domain Controllers
Start Security Domain Requestors	<p>You can start a security domain requestor with a token file, if you have deployed token files for your security installation, or you can start a requestor without a token file if you have implemented security without a token file.</p> <p>You can start the domain requestor without specifying a security token filename.</p> <p>For example:</p> <pre>connect name "ms" discovery "tcp://127.0.0.1:50000" listen "tcp://127.0.0.2:50000" security_token "none"</pre> <p>To start security domain requestor with a token file see Starting a Security Domain Requestor with a Token File.</p>

Creating a Security Policy File

You can create a security policy file to specify security settings across metaspaces, and bind metaspaces to security domains.

Procedure

1. Start the **as-admin** utility:

```
as-admin
```

2. At the as-admin command prompt, enter:

```
create security_policy policy_name "mypolicy/mydomain"
[encrypt <boolean>][validity_days <integer>]
policy_file <policy_filename>
```

For example

```
create security_policy policy_name "mypolicy/mydomain" encrypt true
validity_days 100 policy_file "acme_policy.txt"
```

where:

policy_name

Is an optional parameter that specifies the name of the security policy and security domain that is created. If you do not specify this parameter, a policy named AS-POLICY and a domain named AS-DOMAIN are created.

encrypt

Is an optional parameter that indicates whether the identity for the domain is to be encrypted. Each policy can have one or more domains. The default is encrypt true.

validity_days

An integer that specifies how long the domain ID that the command creates remains valid. The default value is 365 days.

policy_filename

Specifies the name of the policy file that is created.

3. You are prompted to enter a domain password:

```
New domain password [mydomain]:
```

4. Enter a password for the domain.

5. You are prompted to verify the domain password.

```
Verifying - New domain password [mydomain]:
```

6. Re-enter the password to confirm it.

A message appears indicating that the policy was created; for example:

```
Policy created at: acme_policy.txt
```

When using security, the Discovery URL is set on both, the policy and token files. Members using either the policy or the token file should not explicitly specify the Discovery URL in the command line parameters of the as-admin agent.

Edit a Security Policy File

After you have created a policy file for your security domain, you must edit the settings in the file to specify the security configuration for the domain.

For detailed information on editing security policy files, refer to the *TIBCO ActiveSpaces Developer's Guide*.

[Table 37, Security Policy File Sections](#) indicates the sections in the policy file. Some sections are optional and might not need to be modified from the default values or specified in your security policy file. These sections are marked as “optional.”

Security Policy File Sections

Setting	Requirements	Description
Metaspace Access List	Required	Specifies the metaspaces in the security domain and the discovery URL to be used to discover each metaspace. You must specify at least one metaspace and discovery URL. For detailed information on the settings, see <i>TIBCO ActiveSpaces Developer's Guide</i> .
Transport Security	Required	Specifies the type of Transport security used when ActiveSpaces data is transmitted. For detailed information, see <i>TIBCO ActiveSpaces Developer's Guide</i> .
Restricted Transport Access	Required	Specifies whether transport access is restricted to members with specific token identities For detailed information, see <i>TIBCO ActiveSpaces Developer's Guide</i> .

Setting	Requirements	Description
Data Encryption	Optional	<p>Specifies whether tuple data is encrypted in shared memory and when it is persisted on local storage.</p> <p>The default setting, <code>data_encryption=false</code>, specifies that data is not encrypted.</p> <p>For detailed information <i>TIBCO ActiveSpaces Developer's Guide</i>.</p>
Authentication	Optional	<p>Species whether user authentication is required, and if so, what type of authentication is used.</p> <p>The default value, <code>authentication=none</code>, specifies there is no authentication.</p> <p>For detailed information, see <i>TIBCO ActiveSpaces Developer's Guide</i>.</p>
Security Domain Access Control	Optional	<p>Specifies whether access control is required.</p> <p>The default value <code>access_control=false</code>, specifies that there is no access control.</p> <p>For detailed information, see <i>TIBCO ActiveSpaces Developer's Guide</i>.</p>
Access Control Groups	Optional	<p>Allows you to specify what groups or users are granted access to specified to ActiveSpaces operations.</p> <p>The default policy file contains a <code>groups</code> keyword with no groups defined.</p> <p>For detailed information, see <i>TIBCO ActiveSpaces Developer's Guide</i>.</p>
Access Control Permissions	Optional	<p>Allows you to specify which ActiveSpaces operations are permitted for specified groups or users.</p> <p>The default policy file contains a <code>permissions</code> keyword with no permissions defined.</p> <p>For detailed information on setting permissions, see <i>TIBCO ActiveSpaces Developer's Guide</i>.</p>

Validating a Security Policy File

Procedure

- At the `as-admin` command prompt, enter:


```
validate [policy_name <string>] policy_file <string>
```

where

`policy_name` specifies the name of the policy to be validated

`policy_file` specifies the the name of the policy file to be validated.

For example

```
validate policy_name 'mypolicy' policy_file 'policy.txt'
```



On Windows, `validate policy_file` command does not accept back slash ('\') in the directory path

Creating a Security Token

A token is an optional configuration file that can be deployed on nodes that have access to or create secured ActiveSpaces resources. The token is created from the security parameter values set in a specified policy file.

If not used, the keyword “none” is provided for the token file location. In such a case, requestors will trust any controller and these requestors cannot connect to a secured metaspace where transport level authentication is required.

When you create a token, you can specify that it is encrypted: in this case, a requestor can only be started if the password is typed when the node starts.

Procedure

1. Create a policy file and set the policy parameters required for a token.
2. Run the `create security_token` command to create a token file.

The `create security_token` command has the following syntax:

```
define | create security_token domain_name <string>
policy_file <string> [create_identity [common_name <string>]
[encrypt <boolean>][validity_days <integer>]]token_file <string>
```

For a complete description of the `create security_token` command, see [define create security_token](#).

For example:

```
as-admin>
create security_token
domain_name "mydomain"
policy_file "mypolicy.txt"
token_file "mytoken.txt"
```

3. Copy the token file to requestor nodes as needed.
4. Verify OS-level access privileges on the security tokens.

When using security, the Discovery URL is set on both, the policy and token files. Members using either the policy or the token file should not explicitly specify the Discovery URL in the command line parameters of the `as-admin` agent.

Perform Additional Programming Tasks to Process Authentication Requests

To process authentication requests, once you have set up authentication using the ActiveSpaces CLI, you can code a callback routine for client authentication on requestors and develop code to process authentication requests.

Implementing the API's authentication callback is optional. If authentication is required on a metaspace, a default authenticator is always provided, which prompts the user for the username/account-password or keyfile/keyfile-password. If more sophisticated credential feeding implementations are required, the callbacks can be implemented to customize this behavior.

The ActiveSpaces API provides a sample Java program, `ASUserAuthenticator.java`, that demonstrates the use of a callback routine to process user authentication information.

For a general description of user authentication and the `ASUserAuthenticator` sample program, see *TIBCO ActiveSpaces Developer's Guide*.

Validating a Security Token File

Procedure

- At the as-admin command prompt, enter:

```
validate token_file <token_filename>
```

where *token_filename* is the name of the token file to be validated.

For example, enter:

```
validate token_file "mytoken"
```

Resetting the Validity for Policy, Token, or Domain Credentials

OpenSSL can be used to check when a certificate expires. If you have OpenSSL installed on your system, perform the following steps to check the validity period for the certificate in an ActiveSpaces policy or token file.

Procedure

- Extract the certificate information from the policy or token file into a separate text file named `mycert.pem`. The certificate includes the following lines and everything in between them:

```
---BEGIN CERTIFICATE---  
---END CERTIFICATE---
```

- Run the following openssl command on the text file you just created that contains the certificate:

```
openssl x509 -text -noout -in mycert.pem
```

You should see something like the following in the information displayed:

```
Validity  
Not Before: Jul 1 22:26:17 2015 GMT  
Not After : Jun 30 23:26:17 2017 GMT
```



The Not After date is the last day the certificate will be valid.

If you have data encryption enabled, refer to the steps in [Resetting the Validity for a Policy when Data Encryption is Set](#).

Resetting the Validity for a Policy when Data Encryption is Set

When data encryption is enabled, the steps used to reset the validity of a policy, token, or domain credentials are different.

Procedure

- From the old Policy file, copy the private key in a txt file(say `pk.txt`). The private key lies between the following statements:

```
---BEGIN ENCRYPTED PRIVATE KEY---  
---END ENCRYPTED PRIVATE KEY---
```

- Run the following command:

```
openssl req -new -key pk.txt -out newPK.csr
```

- Provide your domain identity password when you are prompted for a password. For example, when you see a prompt like this - Enter pass phrase for `pk.txt`, enter your domain identity password .
- For other prompts, press Enter.

3. Run the following command:

```
openssl x509 -req -days 365 -in newPK.csr -signkey pk.txt -out newSignedPK.crt
```



In the command mentioned, you can change the number of days to suit your requirements.

- a) Here again, provide the domain identity password.
4. In an editor, open `newSignedPK.crt` and copy the content
 5. Paste the content in your policy file between the following statements:

```
-----BEGIN CERTIFICATE-----  
-----END CERTIFICATE-----
```

6. Restart your domain controllers, and run the following command to verify the date:


```
openssl s_client -connect localhost:50000|openssl x509 -noout -dates [localhost:  
50000 is your domain  
controller listening ip and port]
```
7. Recreate token files, and restart all domain requestors.

Manual Process of Changing the Security Policy and Token Files

Updating the security policy and token files when certificates are about to expire is a manual process. Ensure that you go through these steps when the security certificate in a policy or token file is about to expire.

Procedure

1. Generate a new policy file using the same policy and domain names as the old policy file. Be sure to specify the `validity_days` option with the number of days you want the new certificate to be valid for.
2. Edit the settings of the new policy file (metaspace access list, etc.) to match the settings of the old policy file.
3. Generate a new token file based upon the new policy file. Review the settings in the new token file to make sure they match the settings in the old token file.
4. Shut down the cluster and all of its clients.
5. Restart the security domain controller(s) using the new policy file.
6. Restart the rest of the cluster using the new token file.

Starting Security Domain Controllers

Prerequisites

Make sure that the paths to the required system variables are set. Identify a process appropriate for using as a security domain controller. The security domain controller should be a process which is expected to remain running for the life of a metaspace. It is recommended to use `as-agent` for this purpose because the overhead of the domain controller is relatively low. Also, in high availability configurations, where more than one controller with identical policy files is used, you can have one `as-agent` per system configured as a domain controller.



Remember that a security domain requestor is any client process trying to join a secure metaspace that is not already a security domain controller.

Procedure

1. See the *TIBCO ActiveSpaces Installation* document for information on setting the environment variables.
2. Make sure that you have a valid policy file for the domain.
3. Start an **as-agent** as a domain controller:

```
as-agent -security_policy 'mypolicy.txt' -metaspace 'ms'
-member_name 'secure1' -listen 'tcp://127.0.0.1:50000'
```



When you set up a domain controller, make sure that the **-listen** parameter is explicitly set to one (or more) of the URLs IP:PORT included in the discovery URL that is in the policy file. If the discovery URL does not have a port associated with it, the port number of the listen URL must be explicitly set to 50000.

Starting a Security Domain Requestor with a Token File

Procedure

1. Make sure that the paths to the required system variables are set:
2. See the *TIBCO ActiveSpaces Installation* document for information on setting the environment variables.
3. Make sure that you have a valid token file for the domain
4. Start the as-admin utility:

```
as-admin
```

5. Issue the following CLI command to start the requestor node:

```
connect security_token <string> [name <string>] [member_name <string>] [listen
<string>]
```

For example:

```
as-admin> connect security_token 'mytoken.txt' name 'ms' member_name
'client1' listen 'tcp://127.0.0.1:50000'
```

where,

security_token

Specifies a string indicating the name of the token file for the security domain.

name

Specifies a string containing the name of the metaspace that is specified in the Metaspace Access List within the security policy file.

The metaspace name that you use to start the requestor must be the same metaspace name that you used to start the security domain controller.

member_name

Is an optional parameter that specifies a metaspace member name.

listen

Specifies a string indicating the listen parameter for the metaspace.



Do not specify the discovery parameter for the connect command when starting either a security domain controller or a security domain requestor. When you start these nodes, the `connect` command picks up the discovery setting specified in the security policy file or the security token file for the node.

ActiveSpaces Utilities as Windows Services

You can use the `as-service.exe` utility for execution of ActiveSpaces utilities, such as `as-agent`, `ASMM` and the ActiveSpaces Hawk MicroAgent, as Windows services.

The `as-service.exe` utility uses TIBCO Runtime Agent® to run `as-agent`.

To configure TIBCO ActiveSpaces to use `as-service`, on Windows platforms, the following new configuration files are installed in the `AS_HOME/bin` folder.:

as-service.tra

Configuration file for running the Java `as-agent` service.

asmm.tra

Configuration file for running ASMM as a Windows service.

as-hawk-agent.tra

Configuration file for running AS Hawk Microagent as a Windows service.

If needed, you can change the settings in the `.tra` file for the `as-agent` service, `ASMM` service, or `AS Hawk Microagent` service and then install and uninstall the ActiveSpaces services.

You can configure the following properties for `as-agent` running as a service:

- The `AS_HOME` directory (if you need to change the default)
- The `JAVA_HOME` directory
- The ActiveSpaces agent service name
- The heap size for the `as-agent` JVM
- Additional `as-agent` command line arguments, if needed
- Additional path elements, if needed
- The location of the JRE environment

To install and register `as-agent` as a service using the settings in the `as-service.tra` file, enter:

```
as-service --install
```

For additional information, see [The as-service.tra File Example](#) for the text of the `as-service.tra` file, or view the `as-service.tra` file in your installation.

The as-service.tra File Example

The following excerpts from the `as-service.tra` file provide instructions for configuring JVM heap size, `PATH`, and so on, if needed.

The `as-service.tra` file:

```
## Specify the maximum heap size for the as-agent JVM as needed
java.heap.size.max=512M

## Specify the ActiveSpaces agent service name and display name properties
ntservice.name=TIBAS_AGENT_SERVICE
ntservice.displayname=TIBCO ActiveSpaces Agent Service

## Specify additional as-agent command line arguments as needed
## application.args=

## Specify additional path elements if needed
tibco.env.PATH=%AS_HOME%\bin;%AS_HOME%\lib;%PATH%

## Specify (if necessary) the location of the JRE environment.
```

```
##If the AS_HOME environment variable is configured to point to a
##JDK installation then no further changes are necessary
tibco.env.JRE_HOME=%JAVA_HOME%\jre

## Do not change these settings
##
ntservice.binary.path.absolute=%AS_HOME%\bin\as-service.exe
application.start.dir=%AS_HOME%\bin
java.start.class=com.tibco.as.space.agent.AgentConsole
java.start.method=main
java.class.path=%AS_HOME%\lib\as-agent.jar
java.library=%JRE_HOME%\bin\server\jvm.dll
```

The ActiveSpaces Hawk MicroAgent

The TIBCO ActiveSpaces Hawk Microagent is an ActiveSpaces managed application instrumented with the TIBCO Hawk® Application Management Interface (AMI) protocol and which exposes a set of methods that allow the connected Metaspace to be monitored and controlled by TIBCO Hawk tools.

The ActiveSpaces Hawk Microagent communicates with the TIBCO Hawk Agent by using TIBCO Rendezvous as the transport for exchanges of monitored and management data to interact with the Metaspace.

With the TIBCO ActiveSpaces Hawk Microagent, you can perform interactive monitoring through the Hawk display or automate the execution of the methods by creating rulebases to raise alerts and actions performed by the TIBCO Hawk Agents. Refer to the Hawk documentation on the concept of a Hawk Microagent.

You can use the TIBCO ActiveSpaces Hawk Microagent to capture:

- The size, throughput, and growth of a Space monitored by a Hawk rulebase.
- The event on definition or the drop of a Space.
- The joining and leaving of members to the Metaspace as well as member to a Space so alerts and corrective actions can be taken.

Combining the Hawk rules with other TIBCO AMI instrumented applications allows you to build a monitoring and management system for all Spaces and Members of a Metaspace cluster.

Getting Started

The TIBCO ActiveSpaces Hawk Microagent application is packaged and installed as a jar file under the following lib folder:

```
<TIBCO_HOME>/as/<version>/lib/as-hawk-agent.jar
```

Before you start the TIBCO ActiveSpaces Hawk Microagent, ensure that the TIBCO Rendezvous daemon and the Hawk Agent have already been started on the machine. Refer to the TIBCO Hawk documentation for more details.

Start your TIBCO ActiveSpaces application before starting the TIBCO ActiveSpaces Hawk Microagent application to monitor your Metaspace cluster. The Hawk Microagent application primarily collects statistics information for Spaces and Members of your Metaspace by joining the System Spaces as Leech. There is no direct access to the data of the User Spaces.

To start the TIBCO ActiveSpaces Hawk Microagent, the classpath must include the lib folder of both the TIBCO Rendezvous and the Hawk installations. From the command line, it accepts the same set of Metaspace connection arguments as in as-agent, such as Metaspace name, Discovery and Listen URL.

Additionally, it accepts parameters to establish a connection to TIBCO Rendezvous for communications with the Hawk Agent. If no port is specified, the default is port=7474. Refer to the ActiveSpaces documentation for details on the command line arguments for as-agent.

Here is a typical Unix shell script to start the TIBCO Hawk Microagent:

```
#!/bin/sh
export LD_LIBRARY_PATH=<AS_HOME>/lib:<RV_HOME>/lib
java -Djava.ext.dirs=<RV_HOME>/lib:<HAWK_HOME>/lib -jar as-hawk-agent.jar -
metaspace <metaspace>
```

Help information can also be displayed by including with the -usage argument:

```
java -jar as-hawk-agent.jar -usage
```

Once TIBCO ActiveSpaces Hawk Microagent is successfully started, you can use the TIBCO Hawk display to perform interactive monitoring activities on the connected Hawk Agent.

From the Hawk display, gather all Hawk Microagents registered and discovered on the machine. The TIBCO ActiveSpaces Hawk Microagent is displayed as:

```
ASRuntimeInfo: <metaspace>:<version>
```

Selecting the TIBCO ActiveSpaces Hawk Microagent displays all the methods exposed for this Metaspace connection. Note that you can monitor multiple Metaspaces by starting multiple instances of the TIBCO ActiveSpaces Hawk Microagent. Unlike `as-agent`, where you can start multiple instances to perform seeding and operations on User Spaces, you should only have one instance of the TIBCO ActiveSpaces Hawk Microagent per Metaspace connection.

ActiveSpaces Hawk Microagent Methods Overview

The TIBCO ActiveSpaces Hawk Microagent provides the following categories of monitoring methods:

- Space and Member Statistics
- Space and Member Distributions
- Space Operation Throughput at Application Level
- Space Definition Events
- Metaspace Member and Space Member Events
- Console and File Logging Management
- Space Management

From the TIBCO Hawk display, select a TIBCO Hawk Microagent. The microagent displays all the methods exposed for monitoring the Metaspace cluster. You can typically *invoke* for on-demand request to get the result interactively. You can also *subscribe* to the exposed methods specifying the data delivery polling interval. Data delivery interval for Space Definition and Member events are not required as it is instrumented based on ActiveSpaces Event Listener. In the case of a *subscription*, it allows you to create a monitoring rule for building a Hawk rulebase.

Space and Member Statistics Methods

The Space Member statistics methods return the result either on a single row or multiple rows.

The following table lists the space and member statistics methods.

Method	Argument
<code>getSpaceStatisticsByName</code>	SpaceName
<code>getSpaceStatisticsAll</code>	none
<code>getSpaceStatisticsByFilter</code>	SpaceNameFilter, UseRegEx
<code>getMemberStatisticsAll</code>	none
<code>getMemberStatisticsByName</code>	MemberName
<code>getMemberStatisticsByFilter</code>	MemberNameFilter, UseRegEx
<code>getMemberSystemStatistics</code>	MemberNameFilter, UseRegEx

The information reported is equivalent to the statistics information reported by the `as-admin show space` command, which includes Entries, Puts, Gets, and so on. Subscribing on these methods from the Hawk console also allows you to chart the change history of all the reported attributes.

Space Distribution Methods

The Space and Member Distribution methods return the percentage distribution of the data either grouped by the Space Members of a Space or grouped by Spaces of a Member. You can monitor the distribution of data across Members of a Space. Or you can monitor the distribution of Space entries stored grouped by Members.

Method	Argument
getSpaceDistributionByName	SpaceName
getSpaceDistributionAll	none
getSpaceDistributionByFilter	SpaceNameFilter, UseRegEx
getMemberDistributionAll	none

Space Operation Throughput at Application Level Methods

The Space Operation Throughput methods provide the application level throughput based on the changes of the result attributes over the data delivery interval.

For example, it reports the number of Puts per Second by collecting the number of Put operations performed on a Space and divided by the time elapsed over the subscription time interval. You can subscribe and create a rule to create an alert action when the Put throughput spikes or falls below a min/max threshold on a Hawk Rulebases.

Method	Argument
getSpaceThroughputByName	SpaceName
getSpaceThroughputAll	none
getSpaceThroughputByFilter	SpaceNameFilter, UseRegEx
getMemberThroughputAll	none

Space Definition Events Methods

The Definition Event methods provide an event driven subscription service that allows the monitoring of new Space defined or when a Space is dropped or altered. It reports all the Spaces Definitions in multiple result rows when any Space event.

There is one Space Member events method, as shown in the following table.

Method	Argument
onSpaceDefEvent	none

Metaspace Member and Space Member Events Methods

The Metaspace Member and Space Member Events methods provide an event driven subscription service that allows the monitoring of Members joining and leaving the Metaspace or a Space. It reports all the Members of the Metaspace or all the Space Members of a Space along with the attributes of the reported Members when any Member event happens. The additional member attributes also allows

you to monitor when a Metaspaces Member changes the Manager Role or when a Space Member changes its Distribution Role between Seeder and Leech.

The following table lists the Metaspaces Member and Space Member Events methods.

Method	Argument
onMemberEvent	none
onSpaceMemberEvent	SpaceName
onSpaceStateEvent	none
onSpaceStateEventByName	SpaceName

Console and File Logging Management Methods

The Console and File Logging Management methods provide management functionality for console logging and file logging.

The following table lists the Console and File Logging Management methods.

Method	Argument
setMemberConsoleLogLevelAll	LogLevel
setMemberConsoleLogLevelByName	MemberName, LogLevel
setMemberConsoleLogLevelByFilter	MemberNameFilter, UseRegex, LogLevel
setMemberFileLogLevelAll	LogFile, MaxFileSize, MaxRolloverIndex, OpenAppend, LogLevel
setMemberFileLogLevelByName	MemberName, LogFile, MaxFileSize, MaxRolloverIndex, OpenAppend, LogLevel
setMemberFileLogLevelByFilter	MemberNameFilter, UseRegex, LogFile, MaxFileSize, MaxRolloverIndex, OpenAppend, LogLevel

Space Management Methods

The Space Management methods provide space management functionality.

The following table lists the Space Management methods.

Method	Argument
recoverSpace	SpaceName, RecoverWithData, Quorum, KeepFiles

ActiveSpaces Monitoring and Management

TIBCO ActiveSpaces provides a Web-based administrative GUI, called ActiveSpaces Monitoring and Management (ASMM), which connects to one or more metaspaces and monitors the metaspaces, metaspaces members, and connected spaces. Monitoring information includes detailed statistics about members and spaces and performance graphs for space operations.

ASMM provides an ASMM Admin interface and an ASMM Console interface:

ASMM Admin

Controls connections to metaspaces that are to be monitored. Provides a dialog for specifying metaspaces connection parameters, and allows you to connect to metaspaces, and disconnect from them. You can also specify service settings and the port number used for client connections.

For details on the ASMM Admin interface, see the following sections:

- [Starting ASMM and Connecting to a Metaspaces](#)
- [Modifying or Deleting a Metaspaces Connection](#)
- [Connecting and Disconnecting from a Metaspaces](#)
- [Viewing the Members and Spaces Connected to a Metaspaces](#)
- [Using the ASMM Terminal Window](#)
- [Specifying Service Settings](#)
- [Specifying Client Connection Settings](#)

ASMM Console

Provides monitoring functionality for ASMM. Monitoring includes a summary of the currently connected metaspaces, and a set of monitoring views that show various statistics for the connected metaspaces, such as Top Spaces by Entries, Top Spaces by Get, Take, and Put Operations, Average Get Throughput. For more details see *TIBCO ActiveSpaces Monitoring and Management Console Guide*.

Starting ASMM

You need to run the `asmm.jar` to start the ASMM Administrative interface.

Procedure

1. Open a command window.
2. Navigate to the directory where ASMM is installed. For example, `<TIBCO_HOME>\as\<version>\asmm`.
3. Enter the following command:


```
java -jar asmm.jar
```
4. Open a browser, and in the address field, enter the URL for the ASMM Administrative interface. The default value is:


```
http://<hostname>:8680
```



You can change the port number for the ASMM interface by editing the `config.xml` file for the ASMM deployment. The `config.xml` file is located in the `TIBCO_HOME/as/<version>/asmm` directory. To change the port number for the ASMM Admin interface, locate the `<servers>` section in the XML code and change the value for the `<ports>` tag from the default (8680) to the port number that you want to use.

After providing the credentials, the Home page for the ASMM Admin interface is displayed.

Setting up ASMM Credentials

When you first login, the default username/password is admin/admin. The configuration resides in a user name password properties file, `realm.properties`, installed under `TIBCO_HOME/as/<version>/asmm/conf/`.

Procedure

1. Go to `TIBCO_HOME/as/<version>/asmm/conf/realm.properties`.
2. Look for the following code snippet:


```
<username>: <password>[,<rolename> ...]
```
3. Add user names, password, and a comma-separated list of roles by modifying the code snippet shown in step 2.
4. Save the file.

A Sample realm.properties File

```
## This file defines users passwords and roles for a HashUserRealm
## The format is
# <username>: <password>[,<rolename> ...]
## Passwords may be clear text, obfuscated or checksummed. The class
# org.eclipse.util.Password should be used to generate obfuscated
# passwords or password checksums
## If DIGEST Authentication is used, the password must be in a recoverable
# format, either plain text or OBF:.
#
admin: admin,admin
user: password,user
```

ASMM Admin Home Page Reference

ASMM Admin Home Page

The screenshot displays the TIBCO ActiveSpaces Monitoring & Management Admin Home Page. The page is divided into several sections:

- Navigation Menu (Left):** Includes Home, Metaspaces Connections, Service Settings, and Client Connection Settings.
- Monitoring Table (Center):** A table with columns for Service, Status, Monitoring URL, and Action.

Service	Status	Monitoring URL	Action	
ASMM Monitoring Service	started	http://rochide.8686	Stop	
Metaspaces				
Metaspaces	Discovery	Listen	Status	Action
OESS	tcp://127.0.0.1:30011	tcp://192.168.1.192:50002	connected	Disconnect
localms	tcp://10.98.100.292:30012		disconnected	Connect
- System Statistics (Right):**
 - OESS [connected]:**
 - Discovery: tcp://127.0.0.1:30011
 - Listen: tcp://192.168.1.192:50002
 - Manager: 127.0.0.1:30011
 - Members:
 - Servers: 4
 - Clients: 0
 - Hosts:
 - 127.0.0.1: 2
 - 192.168.1.192: 2
 - Agents: 2
 - Admins: 0
 - Hawks: 0
 - Time:**
 - Start Time: 2014-10-28 01:59
 - Uptime: 14 days 7 hours 44 mins
 - Spaces:**
 - Space Definitions: 10
 - Persistence:
 - Share All: 0
 - Share Nothing: 10

ASMM Admin Home Page Metaspace Information

Field	Description
Connection status	The name of the monitored metaspace and the current connection status.
Discovery	The Discovery URL for the connected metaspace.
Listen	The Listen URL.
Manager	The IP address of the cluster member that is currently acting as Manager.
Members	The number of members in the metaspace.
Hosts	The number of hosts running on each host address in the cluster.
Agents	The number of as-agents running in the cluster.
Admins	The number of as-admin processes running in the cluster.
Hawks	The number of AS Hawk Microagents running.
Start Time	The timestamp when the first member joined the metaspace.
Uptime	The duration the metaspace was running since the first member joined.
Space Definitions	<p>The number of spaces defined in the metaspace. Also indicates whether persistence is implemented, and if so, the following:</p> <p>Share All The number of spaces that implement shared-all persistence.</p> <p>Share Nothing The number of spaces that implement shared-nothing persistence.</p>

Creating a New Metaspace Connection

Procedure

1. Open a browser, and in the address field, enter the URL for the ASMM Administrative interface. The default value is:

```
http://<hostname>:8680
```

where *hostname* is the host name or IP address of the host running ASMM.



You can change the port number for the ASMM interface by editing the `config.xml` file for the ASMM deployment. The `config.xml` file is located in the `TIBCO_HOME/as/<version>/asmm` directory. To change the port number for the ASMM Admin interface, locate the `<servers>` section in the XML code and change the value for the `<ports>` tag from the default (8680) to the port number that you want to use.

The Home page for the ASMM Admin interface is displayed.

2. Click **Metaspace Connections**.

3. Click **New**.

The Metaspace Connection dialog is displayed.

Metaspace Connection Dialog

4. Enter the parameters listed in the following table to set up the metaspace connection.

Metaspace	Enter the metaspace name.
Discovery	(optional) Enter a Discovery URL. For example, enter tcp://10.100.200.125:5052. For detailed information on the discovery parameter for metaspace connections, see Table 4, connect Parameters .
Listen	(optional) Enter a Listen URL. For detailed information on the listen parameter for metaspace connections, see Table 4, connect Parameters .
Member Timeout	Specify the amount of time (in milliseconds) to wait for a member to reconnect. The default value is 30000.
Cluster Suspend Threshold	Specify the number of host connections that can be lost after which the membership operations are suspended. The default value is -1 which means that the member operations will never be suspended. When host-aware replication is in use, communication to all members running on a single host must be lost for the host to be considered lost. When host-aware replication is not used, lost communication to any member is considered lost communication to a host. For example, if host-aware replication is not used and two as-agents are running on a single host, if communication is lost to both as-agents, it is considered as communication lost to two hosts.
Allow Space Query	To allow space query on spaces that join the metaspace, select the Allow Space Query check box. Selecting this checkbox enables you to perform queries on the spaces using the ASMM Console interface. Clear this checkbox to prevent contents of user space data from being accessible to the ASMM Console interface. By default, this option is checked.

Secure Connection	To establish a secure connection, select the Secure Connection check box. Provide the following details: <ol style="list-style-type: none"> 1. Authentication: Select an authentication mechanism from the drop-down list. 2. Security Token File : You can browse through your local machine and select a security token file to be used.
Enable	Select the Enable check box to specify that the Metaspace connection should connect automatically on next restart of the ASMM server. The default is set to True.

5. Click **Save**.
6. Click the **Home** tab.

The ASMM Admin Home page is displayed. The metaspace configuration appears in the list of metaspaces.

Connecting to a Metaspace

Procedure

1. On the ASMM Admin home page, click **Connect** in the entry for the metaspace connection. If the connection is successful, a notification is displayed.
2. If the monitoring service is not active and you want to start it, click **Start** in the entry for the ASMM Monitoring Service.
3. To stop the monitoring service, click **Stop**.
4. To go to the ASMM Console interface to monitor the metaspace, click the Monitoring URL link for the monitoring service. For more details about using the Console, refer to *TIBCO ActiveSpaces Monitoring and Management Console Guide*.
5. To disconnect from a connected metaspace, click **Disconnect** in the entry for the metaspace.

Viewing the Members and Spaces Connected to a Metaspace

In the Metaspace Connection window, you can see the member and spaces connected to a metaspace.

Procedure

1. On the ASMM Admin Home page, click **Metaspace Connections**.
2. The Metaspace connections page is displayed.
3. To display the members that have joined the metaspace, click **Members**.
4. To display the spaces that the selected member has joined, click the plus sign (+) next to the member name.
5. To display a list of all of the spaces in the metaspace, click **Spaces**.
6. To collapse the listed of spaces, click the minus sign (-) next to the member name.

Using the ASMM Terminal Window

ASMM provides a terminal window runs the as-admin console for metaspaces that are connected.


Using the as-admin console with ASMM, you can run any as-admin CLI command. All as-admin commands are available for execution except for the **connect** and **disconnect** commands. This is because the as-admin terminal window is context-sensitive to the metaspaces currently selected. You must use the ASMM Administrative interface to connect to or disconnect from a metaspaces. For connecting to a metaspaces, see [Creating a New Metaspaces Connection](#) and for disconnecting from a metaspaces, see [Disconnecting from a Metaspaces](#).

When an as-admin command is active, you can toggle the as-admin display for the command by selecting the row for one of the connected metaspaces. For example, if ASMM is connected to two metaspaces, `metaspaces_one` and `metaspaces_two`, you can enter the **show members** command, and then toggle the display of members for the two connected metaspaces by clicking the row for each metaspaces.

Procedure

1. On the ASMM Admin Home page, click **Metaspaces Connections**.
2. On the Metaspaces Connections page, click **Admin**.


A terminal emulation window appears that runs as-admin.

3. Enter as-admin commands as you would on a native system console.
4. To maximize the console to fill the entire screen, click the  (maximize) icon.

Resetting the Statistics of a Space

Resetting the space statistics on a space helps you reset all the statistics attributes for the space to be 0 except for the number of entries and replicas count.

Procedure

1. On the ASMM Admin Home page, click **Metaspaces Connections**.
2. The Metaspaces connections page is displayed.
3. To display the members that have joined the metaspaces, click **Space**.
4. Click  and select **Reset Reuse Statistics**.
5. From the confirmation window, select **Reset**.
The statistics of the space are reset.

Modifying a Metaspaces Connection

Prerequisites

A metaspaces connection cannot be modified if it is currently connected. If the metaspaces is connected, disconnect the metaspaces. For the steps, see [Disconnecting from a Metaspaces](#).

Procedure

1. From the home page, click **Metaspaces Connections**.
The Metaspaces Connections page is displayed.

The screenshot shows the TIBCO ActiveSpaces Monitoring & Management interface. On the left is a navigation menu with 'Metaspaces Connections' selected. The main area displays a 'New' dialog with a table of metaspaces:

Metaspace	Discovery	Listen	Connection Type	Space Query	Enable
OESS	tcp://127.0.0.1:30011	auto-assigned	plain	allowed	✓
localms	tcp://10.98.100.252:30012	auto-assigned	plain	allowed	✓


Below this is a 'Members' tab showing a list of members:

Member	IP Address	Port	Role
oess-member-two	127.0.0.1	30012	member
oess-member-one	127.0.0.1	30011	manager
asmm.c0a801c0-c352	192.168.1.192	50002	member
as-java-c0a801c0-c350	192.168.1.192	50000	member

2. Click the row for the metaspace whose connection parameters you want to modify.
3. Follow these steps to edit the metaspace connection parameters:
 - a) Click **Edit**.
 - b) On the Metaspaces Connections dialog, modify the connection parameters.
 - c) Click **Save** to save your changes.

Disconnecting a Member

Procedure

1. On the ASMM Admin Home page, click **Metaspaces Connections**.
2. The Metaspaces connections page is displayed.
3. To display the members that have joined the metaspace, click **Members**.
4. Select one or more members that you want disconnected.
5. Click  and select **Disconnect Members**.
6. From the confirmation window, select **Disconnect**.



ASMM itself is connected as a member to a metaspace connection. You cannot disconnect ASMM from the metaspace. Notice that the ASMM member (starts with 'asmm-xxxxx' in the list) cannot be selected for disconnection.

Disconnecting from a Metaspace

Procedure

1. To disconnect from a connected metaspace, on the Home page, click **Disconnect** against the entry for the metaspace.
2. Click **OK** at the confirmation window.
You will get a notification that the metaspace is disconnected.

Deleting a Metaspaces Connection

Prerequisites

A metaspace connection cannot be deleted if it is currently connected. If the metaspace is connected, disconnect the metaspace. For the steps, see [Disconnecting from a Metaspaces](#).

Procedure

1. From the home page, click **Metaspace Connections**.

The Metaspace Connections page is displayed.

The screenshot shows the TIBCO ActiveSpaces Monitoring & Management interface. On the left is a navigation menu with 'Metaspace Connections' selected. The main area is titled 'New' and contains a table of Metaspace connections:

Metaspace	Discovery	Listen	Connection Type	Space Query	Enable
OESS	tcp://127.0.0.1:30011	auto-assigned	plain	allowed	✓
localms	tcp://10.98.100.292:30012	auto-assigned	plain	allowed	✓

Below the table is a 'Members' section with a sub-table:

Member	IP Address	Port	Role
oess-member-two	127.0.0.1	30012	member
oess-member-one	127.0.0.1	30011	manager
asmm-cba801c0-c352	192.168.1.192	50002	member
as-java-c0a801c0-c350	192.168.1.192	50000	member

2. Click the row for the metaspace you want to delete.
3. Follow these steps to edit the metaspace connection parameters:
 - a) Click **Delete**.
 - b) To confirm deletion, click **OK**.

Specifying Service Settings

Using the ASMM Admin interface, you can specify service settings for the Space Statistics Service and for ASMM Console sessions that connect to ASMM to display monitoring data.

Procedure

1. From the ASMM Admin interface Home page, click **Service Settings**.
2. To set the polling interval for the Space Statistics Service, choose a setting from the drop-down list for space settings.

The polling interval specifies how often ASMM polls the connected metaspaces for updates to space statistics. The default value is 5 seconds. You can increase the polling interval to up to 15 seconds.



The polling interval is independent of how often the space statistics are refreshed from a browser session accessing the ASMM Console interface.

3. Specify the following settings:

Space Browser Query Limit	The number of queries allowed for a Space Browser. You can specify a value of up to 10,000 queries. The default value is 2000.
Auto Refresh Interval (seconds)	How often the monitoring display is refreshed on connected ASMM Consoles. You can specify a value of up to 15 seconds.
Session Timeout (seconds)	The time limit after which an ASMM Console session times out. You can specify a value of up to 1800 seconds (30 minutes).



A browser client is an instance of the ASMM Console.

Specifying Client Connection Settings

The Client Connection Settings includes one value: Plain Port. This value specifies the port number used by ASMM Console connections.

Procedure

- In the Plain Port field, enter the port number. The default value is 8686. Using this value, ASMM users connect to an ASMM Console using the following URL:

```
<server_name>:8686
```

Cross-site Replication Concepts

Cross-site replication is a feature that is used for disaster recovery. Using cross-site replication, you can dynamically backup your primary metaspace to a passive metaspace located at a remote site.

The objective of cross-site replication is to maintain two sites in different geographical locations with the same set of data. If a disaster strikes, the remote site or the Disaster Recovery (DR) site should be able to take over for the local or the primary site with very little data being affected.

As you transact with the primary site, you can plan for your data to get backed up on the DR site asynchronously. The replication happens in one direction that is from the primary site to the DR site.

With the help of cross-site replication, you can replicate the following to a DR site:

- PUT and TAKE operations happening on all spaces of a metaspace
- Transactions

Cross-site replication can be incorporated into new or existing deployments of ActiveSpaces using the following components:

- **as-router** executable that is responsible for transmitting PUT and TAKE operations and transactions between sites.
- command-line options in the **as-agent** executable.
- **as-admin** commands.
- Java APIs.

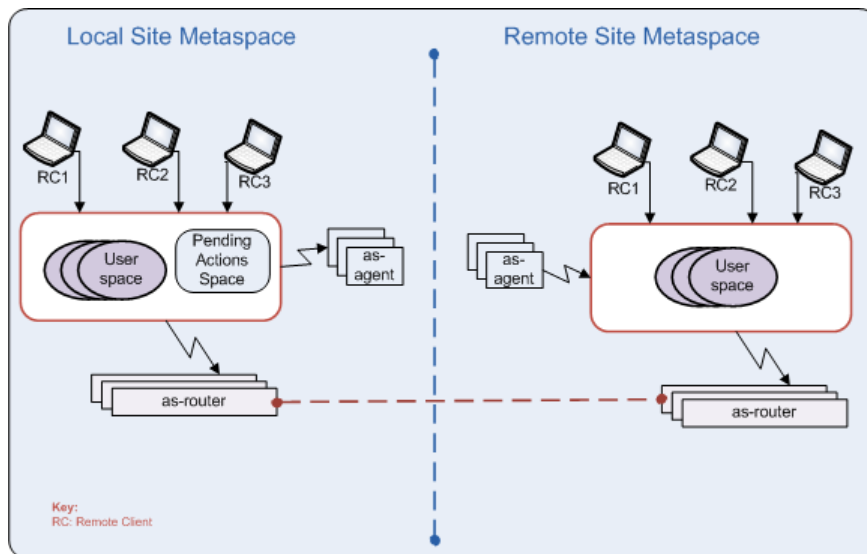
Planning Cross-site Replication

To implement cross-site replication effectively, plan the following:

- Identify a primary site, or the local site. The metaspace on the local site becomes the primary metaspace. This site is usually active. If you have an existing ActiveSpaces system, the existing system would be your primary site. At any point in the discussion of cross-site replication, the terms active and local site are used to refer to the primary site.
- Identify a DR site, or the remote site, that is in a different geographical location. The metaspace on the remote site that is used for the fail-over becomes the secondary metaspace. This site is usually passive. At any point in the discussion of cross-site replication, the terms passive, remote, or secondary site are used to refer to the DR site.
- Both, the primary and secondary metaspace must have the same name, but can have different discovery URLs.
- Both the primary and secondary metaspaces must have the same spaces defined with the same fields and keys.
- Typically, both metaspace clusters are mirror images of each other with the same number of as-agents, as-routers, space replication factor, and persistence modes. But, this can vary between sites as long as the differences between sites meets your disaster recovery requirements. Cross-site messages contain the operation to be performed and the data. No checking is done to ensure the space definitions between sites match. So as long as the space definitions on the DR site can handle the data being sent from the primary site, other properties of the space definitions, such as persistence, replication, or min seeder count, on the DR site can be varied to suit your needs.
- Determine the number of **as-routers** to run which keeps the number of pending operations waiting to be sent to the remote site to a minimum.
- Determine a strategy for switching your clients to using the secondary metaspace and back again as the need arises.

- Determine a strategy for loading and synchronizing data between the primary and secondary metaspaces when both metaspaces are first brought up or when one of the metaspaces is taken down and then brought back up.

Cross-site Replication Between Two Sites



Limitations

- Multi-site replication is not supported.
- Active-Active mode of replication is not supported. Only Active-Passive mode of replication is supported where in the primary site is active and the DR site is passive.
- Security is not supported.
- Cross-site replication is unidirectional. Replication occurs only from the primary site to the DR site.
- Cross-site replication cannot be used with space routing. These two features are mutually exclusive.
- Selection of individual spaces to be replicated is not supported.
- Automatic failover is not supported. Switching over to using your DR site as the primary site is a manual process.
- Space creation on one site is not automatically duplicated on the other site. For cross-site replication to succeed, each space must be predefined on both, the primary and DR site.
- Cross-site replication only replicates PUT and TAKE operations and transactions across sites. It does not automatically create metaspaces, members, spaces, and so on, on the DR site. This process must be done separately for each site involved in cross-site replication.



For spaces with LRU and no persistence defined, when tuples are evicted from the local site, there is no guarantee that the same set of keys are evicted from the remote site.

Local Site Definition

The first step in using cross-site replication is creating a local site definition on each metaspaces involved in cross-site replication.

To create a local site definition, you must do two things:

- Assign a unique site name to your local site
- Enable cross-site routing for the local site

The first member of a metaspaces that is started must create a local site definition. The local site definition is associated with the metaspaces, so any future members that join the metaspaces are

automatically members of the named site. If subsequent members also create a local site definition, the local site definition of those members is ignored and the original local site definition is used.



If you are a new user of cross-site replication, it is recommended that you make **as-agent** or **as-router** the first member of the metaspace.

ActiveSpaces facilitates defining creating a local site definition when the following are run as the first metaspace member:

- **as-agent**
- **as-router**
- custom Java applications which uses the ActiveSpaces Java API



The first metaspace member started also acts as the metaspace membership manager. If the first metaspace member exits, the membership manager role is shifted to another metaspace member thus incurring some delay and overhead in the system. Ideally, the first metaspace member would be one which is expected to run for the life of the metaspace.

Starting as-agent as the First Metaspace Member

The easiest and recommended way to define your local site is to start an **as-agent** as the first metaspace member.

To use an **as-agent** to define your local site specify the following command line option when starting the **as-agent**:

```
-site_name <string>
```

Use the `-site_name` command line option to define your local site using the specified name. This automatically enables cross-site routing for the metaspace.



It is not recommended to have an **as-agent** which acts as a 'leech-only' proxy for remote client connections as the first metaspace member. This is because sometimes these proxies are restarted which would cause the metaspace manager role to shift to another metaspace member.



Cross-site replication is not supported for .NET as-agents.

Starting as-router as the First Metaspace Member

To facilitate using cross-site replication while not having to change your existing ActiveSpaces deployment, you can start an **as-router** as the first metaspace member.

To use an **as-router** to define your local site specify the following command line option when starting the **as-router**:

```
-site_name <string>
```

Use the `-site_name` command line option to define your local site using the specified name. This automatically enables cross-site routing for the metaspace.



as-routers are leech members of the metaspace which provide the routing capabilities between the two metaspaces involved in cross-site replication. There may be cases where you want to restart an **as-router** to re-establish communication with the other metaspace. Therefore, as-routers are not the best choice for a first member of the metaspace for the same reason that as-agents running as 'leech-only' proxies are not the best choice. See the section [The as-router Executable](#) for more information about as-routers.

Starting a Custom Java Application as the First Metaspace Member

For those systems which have written their own custom applications using the ActiveSpaces Java API to replace the use of as-agents, the custom application must be updated to do the following:

- define the local site

- enable the metaspace member as a system seeder

To define the local site, use the Java API to:

1. Create a `LocalSiteDef` object before your code connects to the metaspace.
2. Specify a site name using `LocalSiteDef.setName(<site name string>)`.
3. Enable cross-site routing using `LocalSiteDef.setRouted(true)`.

To enable the metaspace member as a system seeder:

1. Use the `MemberDef` object that your application has already created
2. Before connecting to the metaspace, enable the member to act as a system seeder using `MemberDef.setSystemSeeder(true)`

Finally, update your code which connects to the metaspace:

1. Use the `Metaspace` object that your application has already created.
2. Connect to the metaspace using `Metaspace.connect(String, MemberDef, LocalSiteDef)`.

Metaspace members acting as system seeders provide resources (seed) to the internal system space used in cross-site replication. This is for storing operations waiting to be sent to or received from a site involved in cross-site replication. Typically each **as-agent** acts as a system seeder. When not using as-agents, you must enable your custom seeding applications to also seed this internal cross-site specific space.

Remote Site Definition

Each site involved in cross-site replication considers the other site to be the remote site. Before cross-site replication begins, each site must define the remote site from their own perspective.

Use the following **as-admin** command to define the remote site:

```
create | define site <string> [router_discovery <string>] [suspend <boolean>]
```

Using the `create site` command, specify the name of the other site involved in cross-site replication from the perspective of the site you are actively configuring. To know more about using the `create site` command, see [create site](#).

Using The `router_discovery` Setting

The cross-site replication feature is designed for active-passive replication where the primary site is considered the active site and the DR site is considered the passive site. The `router_discovery` setting must be used when you are defining the remote site on the primary site. The `router_discovery` setting specifies the listen URLs of each **as-router** running on the remote site. If a `router_discovery` setting is not provided while defining the remote site on the primary site, the messages do not get routed to the DR site.



Since active-active replication is not supported, to prevent accidentally trying to replicate data from the remote site to the primary site, do not specify the `router_discovery` setting while defining the remote site on the DR site itself. However, writes to the DR site should never be allowed unless the primary site has gone down and you have failed over to the DR site. Reads on the DR site are allowed.

Using the `suspend` Setting

The `suspend` setting is an optional setting that acts as a toggle. This setting indicates whether or not to suspend the messages being routed to the remote site being defined. By default, the routing of messages begins immediately. Depending on how your application initially loads its data, or whether you are running in a fail over state, you may want to consider defining your remote site with routing initially suspended. You can choose to suspend routing of messages to the remote site being defined

and have the messages saved to the internal Pending Actions Space until the remote site is manually resumed.

Using the suspend setting can be especially useful to synchronize data from your DR site back to your primary site, if you need to temporarily take down your primary site and use your DR site as your main site until your primary site comes back online. To fail over to your DR site and retain any new data updates for later synchronizing back to your primary site, on your DR site update the remote site definition (for your primary site) as follows before switching your clients over to using the DR site:

1. Set the `router_discovery` setting to contain the listen URLs of the as-routers running on the primary site.
2. Set the `thesuspend` setting to `true`.

When clients start using the DR site as the main site, any `PUT` or `TAKE` operations and transactions are stored internally on the DR site. When the primary site is brought back online:

1. Prevent clients from making any changes to the DR site.
2. On the primary site, suspend the DR site.
3. On the DR site, resume the primary site.

Resuming the primary site from the DR site causes the DR site to synchronize the saved operations back to the primary site. Once the synchronization is complete do the following:

1. On the DR site, suspend the primary site (or remove the '`router_discovery`' setting of the primary site).
2. On the primary site, resume the DR site.
3. Switch clients back to using the primary site.



Prior to using this procedure, you must ensure that you have adequate memory on your seeders to store the operations waiting to be synchronized back to the primary site. This procedure is intended for scenarios where the primary site is down for short periods only. You should always have a backup procedure in place for synchronizing data between sites using your persisted data stores or by some other means, if not using persistence.

Example of Suspended Routing

Consider the ActiveSpaces example, ASPersistence, which loads data into ActiveSpaces when it is first started. To initialize your primary site for cross-site replication with shared-all persistence, both the **as-router** executable and the ASPersistence example must be started on the primary site before you have initialized your remote site. Now, you have the following scenario:

- the remote site is not yet running and waiting for connections from the local site
- the ASPersistence example has started and initially loads data into the space on your primary site

In such a scenario, the cross-site messages to your remote site do not succeed and are lost as a connection has not yet been established to your remote site.

If you intended for the data that was initially loaded to be replicated to your remote site, you have two options:

- Ensure the remote site is up and running before starting the primary site.
- When creating a remote site definition on your primary site, specify `suspend` to be `true`. This causes the initially loaded data from the ASPersistence example to be stored until sending to the remote site has been resumed.

Using option 2 implies that on the primary site you have started the **as-router** executable and created your remote site definition in suspended mode prior to starting the ASPersistence example (or any other application which stores data that you expect to be replicated across sites). When a remote site definition is created in the suspended mode, the cross-site routing messages are queued up until the site is later resumed. When a site is resumed, any queued messages are then automatically sent to the site.

Remote Site Definition - Alternative

An alternative to starting **as-admin** and creating a remote site definition using the **create site** command is to use the `-input` command line option of the **as-router** executable. Using the `-input` option you can specify the path and file name of a file which contains **as-admin** commands. These commands are executed by **as-router** after it finishes its initialization. So, if `DRSiteDefintion.txt` contains the **as-admin** command to define your remote site, you can specify `-input "DRSiteDefintion.txt"` as an option when starting the **as-router** executable.



If starting more than one **as-router** executable from a script which specifies the `-input` option, only the first router started creates the remote site definition. For subsequent executions/calls of **as-router**, an error is returned as the DR site is already defined by the first execution of **as-router**. The **as-router** executables ignore errors returned from running the **as-admin** commands and continues to run as usual.

The as-router Executable

The **as-router** executable provides the routing of transactions and space **PUT** and **TAKE** operations from the primary metaspace to the DR (disaster recovery) metaspace.

Multiple **as-routers** can be run on each site. Operations to be handled by **as-routers** are distributed to the **as-routers** in a round-robin fashion on both the primary and DR sites. Typically, you would run the same number of **as-routers** on each site. The idea is to run enough **as-routers** so as to keep the number of operations waiting to be sent to the DR site to a minimum. The number of operations waiting to be sent to the DR site can be monitored using the **as-admin** command **show local site**. The `Pending route actions` field indicates the number of operations in the internal space that are waiting to be sent to the DR site.

Operational Flow

The following is an example of the flow of a put operation when cross-site replication is used.

On the primary site, a client initiates the `PUT` operation of some data into a space. After the data is put and replicated into the appropriate space on the primary site, the `PUT` operation is stored into an internal routing space on the primary site and the client is unblocked. The `PUT` of data into the internal routing space triggers an `as-router` to pick up and asynchronously send the operation to the DR site. When the DR site receives the cross-site message, it sends an acknowledgment back to the primary site which then causes the primary site to remove the operation from the internal routing space. Back on the DR site, the operation is directed to an `as-router` which then performs the operation on the DR site.

This same basic flow is used for take operations and transactions. In the case of transactions, the entire transaction is stored to the internal routing space after the transaction has been committed on the primary site. Even though the transaction may contain multiple operations, you only see one entry for the entire transaction stored in the internal routing space.

Transmission Errors

Each operation put into the internal routing space is assigned a sequential log number. If a transmission error occurs while sending an operation to the DR site, the operation, and its log number, is put back into the internal routing space for resending. The sequential log number of an operation is used on the DR site to ensure that an operation with a smaller log number is not processed if an operation on the same key was already processed and the previously processed operation had a larger log number.

Seeding the Internal Routing Space

`as-routers` run as leech clients of the metaspace and do not seed any user spaces or the internal routing space. The C and Java `as-agents`, by default seed the internal routing space. If you are using your own custom built applications to seed your user spaces, those applications must be updated to also seed the internal routing space used for cross-site replication. See the section [Starting a Custom Java Application as the First Metaspace Member](#) for more information on how to configure your applications to seed the internal routing space.

Starting `as-router`

`as-router` is provided as a native executable only.

Procedure

1. Ensure the ActiveSpaces `/bin` directory is defined in the path environment variable. Go to the ActiveSpaces `/bin` directory.
 - Microsoft Windows default location: `<TIBCO_HOME>\as\<version>\bin`
 - UNIX/Linux default location: `/opt/tibco/as/<version>/bin`
2. Start `as-router` as follows:

```
as-router -router_listen <url>
```

Where:
`<url>` is a string of the form `tcp://<ip_addr>:port` which specifies the IP address and port the `as-router` listens on for connections from other `as-routers`.

as-router Command Line Parameters

The command line parameters for the **as-router** executable can be seen by running:

```
as-router -help
```

The output of the **help** command includes the default values used if no parameter is provided. Except for the two cross-site replication related parameters, the rest of the parameters are also used by the **as-agent** executable. So, to see specific information about any of the non-router specific parameters, see the [as-agent Command Parameters](#) section in the *TIBCO ActiveSpaces Administration* guide. The following is the output of running **as-router -help**:

Required Parameters:

```
-router_listen <url>
```

The following parameter is required if **as-router** is started as the first member of the metaspace:

```
-site_name <site_name>
```

Optional Metaspace Connection Parameters:

```
-metaspace <metaspace_name>      default ms
-discovery <url> / <url_list>     default tibpgm
-listen <url>                      default tcp
-remote_listen <url>
-member_name <member_name>
-worker_thread_count <count>      default 32
-rx_buffer_size <size>            default 2mb
-connect_timeout <ms>             default -1 (no timeout)
-member_timeout <ms>              default 30000
-client_timeout <ms>              default 10 min
-cluster_suspend_threshold         default -1
```

Optional Security Parameters:

```
-security_policy <policy_path>
-security_token <token_path>
-identity_password <password>
-authentication_domain <domain_name>
-authentication_username <user_name>
-authentication_keyfile <file_path>
-authentication_password <password>
```

Optional Logging Parameters:

```
-log <file_path>
-log_debug <log_level>             default 3 (INFO)
-log_limit <limit>                 default -1 (no limit)
-log_count <count>                 default 1
-log_append <boolean>             default true
-debug <log_level>                 default 3 (INFO)
-advisory_level <level>           default 3 (INFO)
```

Optional Monitoring Parameters:

```
-monitor_system <boolean>         default false
```

Optional Script File Parameters:

```
-input <script_path>
```

Discovery url format:

```
tcp://interface:port;interface2:port2;interface3:port3
tibpgm://dport/interface;multicast/key1=value1;key2=value2;key3=value3
```

Listen url format:

```
tcp://interface:port
```

Remote listen url format:

```
tcp://interface:remote_listen_port
```

Router listen url format:

```
tcp://interface:router_listen_port
```

as-admin Commands for Cross-site Replication

The **as-admin** utility offers some commands for cross-site replication.

To see a list of all **as-admin** commands related to cross-site replication, use the following command:

```
help site
```

Enter **help -long** before the name of a specific **as-admin** command to display detailed information about that command. For example, the following command lists the syntax, command description, and parameter descriptions of the **alter site** command.

```
help -long alter site
```

Administering the Primary Site

When cross-site replication is used, the following **as-admin** commands can be used to administer the primary site:

- **show sites** - Used to display detailed information about both the primary and DR site.
- **show [local] site** - Used to display the primary site's settings.

[as-admin Commands for Cross-site Replication](#) section contains detailed information on all of the cross-site related commands supported by **as-admin**.

Administering the DR Site

Once a DR site is defined, the following **as-admin** commands can be used to administer the DR site:

- **alter site** - Used to change the `router_discovery` or `routing` state of the DR site.
- **drop site** - Used to remove the DR site definition from a metaspace.
- **suspend site** - Used to suspend sending cross-site replication messages to the DR site.
- **resume site** - Used to enable ending of cross-site replication messages to the DR site.
- **show sites** - Used to display detailed information about both the primary and DR site.
- **show remote site** - Used to display the DR site's settings.
- [The as-admin Commands for Cross-site Replication](#) section contains detailed information on all of the cross-site related commands supported by **as-admin**.

Deploying Cross-site Replication

Before you deploy cross-site replication, you should also consider the following:

- How to set up each site so that the same spaces are defined on each site.
- How to synchronize the initial loading of data into the spaces on both sites.
- What strategy is used to have clients switch over to using the DR site, if a disaster occurs and the primary site is down.
- What strategy is used to synchronize the data and bring the primary site back online after the DR site has been running in its place.

Defining Spaces

Cross-site replication works across all spaces in a metaspace. If data is written to a space on the primary site, that same space must already be created on the DR site for cross-site replication to succeed.

If your current design has clients or seeders dynamically creating spaces, you have to revisit your strategy to ensure all spaces are created on the DR site before cross-site replication is enabled. It is not necessary that the space definitions on the DR site must exactly match the space definitions on the

primary site. What is important is that the spaces have the same name and can hold the same data. In the space definition on the DR site, you can vary other space definition settings to suit your needs such as: min seeder count, replication count, and so on.

Admin Command Scripts

If you are not already familiar with Admin script files, see the [Using a Script File to Pass Arguments](#) section in the *TIBCO ActiveSpaces Administration* guide. **as-admin**, **as-agent**, and **as-router** can all be used to run Admin script files using the following command line argument:

```
-input <filename>
```

When deploying cross-site replication, Admin command scripts can be useful for:

1. Defining all of the spaces of your metaspace
2. Creating your remote site definition

For example, a file containing the following **as-admin** commands can be used on your DR site to create a remote site definition for a site named 'primarySite' and create the space used by the ASOperations example.

```
create site 'primarySite'

create space name 'myspace' ( field name 'key' type 'integer' nullable false, field
name 'value' type 'string' nullable true, field name 'time' type 'datetime'
nullable true) key (type 'hash' fields ('key'))
```

Notice that `router_discovery` is not specified as a part of the `create site` command on the DR site.

Setting Up Cross-Site Replication on the DR Site

To use cross-site replication, you may find it is easier to set up your DR site first as you do not configure the `router_discovery` setting of the remote site definition on the DR site. On the primary metaspace, the `router_discovery` setting is composed of the `router listen` URLs for all of the **as-router** executables running on the DR site. So, to configure the primary site, you must know how the **as-routers** have been started on the DR site.

Procedure

1. Start your first metaspace member using the additional command line option: `-site_name <DR site name>`
2. Start your ActiveSpaces seeders.
3. Ensure your spaces are defined.
4. Ensure the appropriate data is pre-loaded into your spaces.
5. Start your **as-router** executables.
6. Determine the `router_discovery` setting to be used on your primary site from the `router_listen` settings of all of the `as-routers` started on your DR site.
7. Create a remote site definition for your primary site. Do not specify a `router_discovery` setting in your remote site definition.

Setting Up Cross-Site Replication on the Primary Site

Procedure

1. Start your first metaspace member using the additional command line option: `-site_name <primary site name>`

2. Start your ActiveSpaces seeders.
3. Ensure your spaces are defined.
4. Ensure the appropriate data is pre-loaded into your spaces.
5. Start your as-routers.
6. Create a remote site definition for your DR site. Use the `router_discovery` setting you determined in step 6 under [Setting Up Cross-Site Replication on the DR Site](#).
7. Wait at least 90 seconds for the as-routers on the primary site to connect to the as-routers on the DR site. If the as-routers on the DR site are up and running, you can see the connections established in the console output of the as-routers on the primary site. However, if they are not up and running, it takes about 90 seconds for the connection attempt to timeout and the status to get updated.
8. Use the `as-admin` command `show remote site` to ensure the as-routers on the primary site have connected to the as-routers on the DR site. If the as-routers on the primary site cannot connect to the as-routers on the DR site after 90 seconds, the `local site state` displayed by the `show remote site as-admin` command is set to `offline`.
9. Once the as-routers on the primary site have connected to the as-routers on the DR site, start your ActiveSpaces clients.

The as-routers on your primary site do not try to connect to the as-routers on your DR site until you create a remote site definition for the DR site on your primary site and specify the `router_discovery` setting to use. If an initial connection to the DR site cannot be established by the as-routers, the `local site state`, displayed by the `as-admin` command `show remote site`, is set to `offline` and the as-routers periodically try to establish a connection. After a connection has been established, the `local site state` is set to `online` and cross-site data replication begins.



On the primary site, you can also use Admin command scripts to create spaces and define your remote site.

Monitoring Cross-site Replication

To monitor the status of cross-site replication, on your primary site run the following `as-admin` commands:

- `show local site` to monitor the number of operations waiting to be sent to the DR site.
- `show sites` or `show remote site` to monitor the status of the connection to the DR site.

The number of operations waiting to be sent to the DR site should be kept to a minimum. If the number of operations reported in the `Pending Route Actions` field of the `show local site` command seems unusually high or starts to grow, check to ensure the DR site is still online by running the `show sites` command.



It is important to make sure that the size of the internal space used for cross-site replication does not grow large enough that you run out of memory on your system. Be sure to have a strategy in place for the scenario where your DR site is down for an extended period and you have to disable cross-site replication during that period. Your strategy should include how to ensure that your data is synchronized between your primary and DR sites before bringing the DR site back online.

If the DR site is not online or if too many network errors occur when communicating with the DR site, the site is automatically placed in the offline state. Unless you have manually suspended the DR site, ActiveSpaces automatically detects when the DR site comes back online and continues replicating operations.



Pending operations held in the internal space are not persisted. If the primary site goes down, any pending operations waiting to be sent to the DR site will be lost.

An Example of Simple Cross-site Replication

The following steps are provided as a simple example of how to start up cross-site replication for the ASOperations C example which ships with ActiveSpaces.

On the DR site (assuming an IP address of 10.0.1.10):

1. Start an **as-agent**:

```
AS_HOME/bin/as-agent -discovery "tcp://10.0.1.10:50000" -site_name
"drSite"
```

2. Start an **as-router**:

```
AS_HOME/bin/as-router -discovery "tcp://10.0.1.10:50000" -router_listen "tcp://
10.0.1.10:60000"
```

3. Start **as-admin**:

```
AS_HOME/bin/as-admin -discovery "tcp://10.0.1.10"
```

4. Using **as-admin**, define your space:

```
create space name 'myspace' ( field name 'key' type 'integer' nullable false,
field name 'value' type 'string' nullable true, field name 'time' type
'datetime' nullable true) key (type 'hash' fields ('key'))
```

5. Using **as-admin**, create your remote site definition:

```
create site 'primarySite'
```

On the Primary site (assuming an IP address of 10.0.1.15):

1. Start an **as-agent**: `AS_HOME/bin/as-agent -discovery "tcp://10.0.1.15:50000" -site_name "primarySite"`

2. Start an **as-router**: `AS_HOME/bin/as-router -discovery "tcp://10.0.1.15:50000" -router_listen "tcp://10.0.1.15:60000"`

3. Start **as-admin**: `AS_HOME/bin/as-admin -discovery "tcp://10.0.1.15:50000"`

4. Using **as-admin**: create your remote site definition:

```
create site 'drSite' router_discovery "tcp://10.0.1.10:60000"
```

5. Start ASOperations: `AS_HOME/examples/c/ASOperations -discovery "tcp://10.0.1.15:50000"`

On Both Sites

After you have ensured that the **as-router** connections from the primary site to the DR site have succeeded, you can then use ASOperations as normal and see the data replicated to the DR site.

Shared-All Persistence Usage

1. Cross-site replication only replicates PUT and TAKE operations and transactions across sites. On the DR site, it does not automatically create metaspaces, members, spaces, and so on. This is up to you to handle appropriately.
2. The space definition on the DR site need not exactly match the space definition on the primary site. Only the space name and data field related settings should match. For example, if you expect to failover to the DR site and perform the same queries as you would on the primary site, then the indexes should match. Other space definition settings, such as persistence and minimum seeder count, can be varied on the DR site according to your needs.
3. Cross-site replication occurs for all spaces within a metaspace. You cannot choose individual spaces to be replicated.

4. Space routing cannot be used with cross-site replication.

It is not required that persistence must be run on both the primary and DR sites. If you are running shared-all persistence on both your primary and DR sites, keep the following in mind:

1. Each site should have an independent persister callback implementation and external persistent storage medium.
2. Remember space routing cannot be used with cross-site replication. The `setPersister()` calls are not routed. Ensure that you call `setPersister()` on both the sites.
3. Run the persister callback client on each site.

Alternatively, you can run shared-all persistence only on your primary site without also running it on your DR site. For this scenario, create a space definition on your DR site without setting the persistence type and persistence policy. With cross-site replication, space definitions between the primary site and DR site are not compared. Only the space name and data fields need to match. Other space definition settings, such as persistence, can differ between sites.