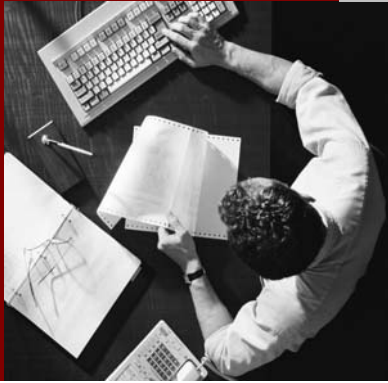


CUSTOMER



Installation Guide

# SAP NetWeaver Standalone Engine Search and Classification TREX 7.0 Multiple Hosts

Target Audience

- System administrators
- Technology consultants

Document Version 1.4 – October 20, 2017

© 2017 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company. The information contained herein may be changed without prior notice.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

Please see <http://www.sap.com/corporate-en/legal/copyright/index.epx> for additional trademark information and notices.

# Important Disclaimers and Legal Information

## Coding Samples

Any software coding and/or code lines / strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, unless damages were caused by SAP intentionally or by SAP's gross negligence.

## Accessibility

The information contained in the SAP documentation represents SAP's current view of accessibility criteria as of the date of publication; it is in no way intended to be a binding guideline on how to ensure accessibility of software products. SAP in particular disclaims any liability in relation to this document. This disclaimer, however, does not apply in cases of willful misconduct or gross negligence of SAP. Furthermore, this document does not result in any direct or indirect contractual obligations of SAP.

## Gender-Neutral Language

As far as possible, SAP documentation is gender neutral. Depending on the context, the reader is addressed directly with "you", or a gender-neutral noun (such as "sales person" or "working days") is used. If when referring to members of both sexes, however, the third-person singular cannot be avoided or a gender-neutral noun does not exist, SAP reserves the right to use the masculine form of the noun and pronoun. This is to ensure that the documentation remains comprehensible.

## Internet Hyperlinks

The SAP documentation may contain hyperlinks to the Internet. These hyperlinks are intended to serve as a hint about where to find related information. SAP does not warrant the availability and correctness of this related information or the ability of this information to serve a particular purpose. SAP shall not be liable for any damages caused by the use of related information unless damages have been caused by SAP's gross negligence or willful misconduct. All links are categorized for transparency (see: <http://help.sap.com/disclaimer>).






## Open Source Software and Third Party Components

Please refer to <https://scn.sap.com/docs/DOC-42044> for information respecting open source software components made available by SAP as part of SAP NetWeaver and any specific conditions that apply to your use of such open source software components. Please refer to <https://scn.sap.com/docs/DOC-42045> for information relating to SAP's use of third party software with or within SAP NetWeaver.

## Typographic Conventions

Type Style	Description
<i>Example Text</i>	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options.  Cross-references to other documentation
<b>Example text</b>	Emphasized words or phrases in body text, graphic titles, and table titles
EXAMPLE TEXT	Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text, for example, SELECT and INCLUDE.
Example text	Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
<b>Example text</b>	Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.
<Example text>	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
EXAMPLE TEXT	Keys on the keyboard, for example, F2 or ENTER.

## Icons

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax

Additional icons are used in SAP Library documentation to help you identify different types of information at a glance. For more information, see *Help on Help → General Information Classes and Information Classes for Business Information Warehouse* on the first page of any version of *SAP Library*.

## Content

Distributed TREX Systems (Multiple Host Installation) .....	1
Naming Conventions .....	2
Required Documentation .....	3
Fundamentals .....	4
TREX Architecture .....	4
Java Client and ABAP Client .....	5
Web Server with TREX Extension .....	5
RFC Server .....	5
Name Server .....	5
Queue Server .....	6
Preprocessor .....	6
Index Server .....	7
TREX Instances and the TREX System .....	7
Distributed TREX Systems .....	8
Server Tasks .....	8
Master, Slave, and Backup Index Servers .....	8
Master and Backup Queue Servers .....	9
Master and Slave Name Servers .....	9
Preprocessor Modes .....	9
Master and Slave Index .....	10
Connection to the Application .....	10
Data Storage .....	12
Supported Systems .....	14
Systems with Master and Slave Index Servers .....	15
Systems with Master, Backup, and Slave Index Servers .....	18
Summary: High Availability .....	21
Global File System and TREX Instances .....	22
Hardware, Software, and Other Requirements .....	25
Constraints .....	29
Planning .....	30
Setting Up a Distributed System .....	31
Checklists .....	32
Preparing for Centralized Data Storage .....	33
Creating a Central TREX Data Directory .....	33
Only UNIX: Mounting the Central TREX Data Directory .....	34
Only Windows: Defining the Network Drive .....	34

Activating the Configuration Clones for Server Blades .....	35
Landscape Configuration .....	37
Defining a New Landscape.....	37
Adding a Host .....	38
Defining the Roles of Hosts .....	38
Configuring Centralized Data Storage.....	40
Checking and Activating the Configuration .....	42
Example Configurations .....	42
Systems with Master and Slave Index Servers .....	42
Systems with Master, Backup, and Slave Index Servers .....	46
Configuration of the RFC Connection .....	51
Creating a SAP System User for the TREX Admin Tool (Stand-Alone) .....	52
Determining the SAP System Connection Information .....	53
Configuring the RFC Connection in the TREX Admin Tool.....	54
Configuring the HTTP Connection .....	56
Information on Breakdown of Master Servers.....	57
Delta Index and Index Replication Configuration .....	58
Delta Index Configuration.....	59
Delta Index .....	59
Activating the Delta Index.....	60
Integrating a Delta Index into the Main Index.....	60
Index Replication Configuration .....	63
Index Replication Process .....	63
Triggering Index Replication.....	66
Controlling the Replication Load .....	69
Configuring Topicality of Search Results .....	69
Changing a Distributed System .....	70
Adding and Removing Hosts.....	70
Removing a Host .....	71
Adding a Host .....	72
Changing Hosts.....	73
Adding a TREX Index Server .....	73
Optimizing the Landscape Using the Reorg Function in the TREX Admin Tool .....	74
Reorganization of the TREX System Landscape.....	75
Adding a Master Host.....	77
Adding a Backup Host.....	79
Adding a Slave Host .....	79
Removing a Backup Host.....	80
Removing a Slave Host.....	81

Replacing a Backup or Slave Host.....	81
Changing Index Assignments .....	82
Changing Queue Assignments .....	82
Allowing Searching on Master Indexes.....	83
Changing Default Directories for Indexes, Snapshots, or Queues .....	85
Distributed Preprocessing of Documents .....	86
Fundamentals .....	87
Preprocessing Flow .....	87
Distributing Preprocessing.....	89
Load Distribution and Performance.....	89
Number of Preprocessors and Preprocessor Threads.....	90
Preprocessor Threads and Queue Server Pool Size .....	92
Configuration.....	94
Configuration Recommendations .....	94
Setting Up Distributed Preprocessing .....	95
Example Configuration.....	97
Increasing the Number of Preprocessors.....	98
Appendix.....	99
Information on Stopping/Starting Distributed Systems .....	99
Starting the TREX Admin Tool .....	100
Configuring Queue Parameters .....	100
Changing Java Client Parameters .....	101



# Distributed TREX Systems (Multiple Host Installation)

## Purpose

Search and Classification (TREX) consists of a client component and a server component. The server component is based on a flexible architecture that allows a distributed installation. A distributed system has the following advantages:

- Load distribution  
You can distribute the search and indexing load among several hosts.
- High availability  
You can make searching and indexing highly available.

This guide explains how to plan and implement a distributed system. It is aimed at technology consultants.

The guide is structured as follows:

- [Naming Conventions \[Page 2\]](#) contains information on the naming conventions used in this guide.
- [Required Documentation \[Page 3\]](#) lists the documentation that you need to implement a distributed system.
- [Fundamentals \[Page 4\]](#) contains information on the TREX architecture and basic information on distributed systems. You need this information to plan a distributed system. Read this information before you begin to implement your distributed system.
- [Setting Up a Distributed System \[Page 31\]](#) and [Delta Index and Index Replication Configuration \[Page 58\]](#) describe how to implement a distributed system.
- [Changing a Distributed System \[Page 70\]](#) describes changes that you can make to your system after the installation.
- [Distributed Preprocessing of Documents \[Page 86\]](#) describes how to distribute the preprocessing of documents among several hosts. This section is relevant if you want to index documents whose preprocessing takes up a lot of time and system resources. This can be the case if you want to index large PDF files.
- The [appendix \[Page 99\]](#) contains information on stopping and starting a distributed system. It also contains information on starting the TREX admin tool and changing the queue parameters and the Java client parameters.





## Naming Conventions

The following conventions are valid for this documentation.

### Terminology

Term	Meaning
TREX instance	One installation of the TREX server software
TREX host	Host on which the TREX server software is installed
Server	Program that offers services (such as an index server or queue server)
Master host	Host on which a master index server is running
Slave host	Host on which a slave index server is running
Backup host	Host on which a backup index server is running

### Variables

Variable	Meaning
<SAPSID>	System ID in uppercase letters
<sapsid>	System ID in lowercase letters
<TREX_DIR>	Installation directory for a TREX instance. The path to the directory is: <ul style="list-style-type: none"> <li>On UNIX /usr/sap/&lt;SAPSID&gt;/TRX&lt;instance_number&gt;</li> <li>On Windows &lt;disk_drive&gt;:\usr\sap\&lt;SAPSID&gt;\TRX&lt;instance_number&gt;</li> </ul>
User <sapsid>adm	Operating system user that you log on with to administrate TREX.
User SAPService<SAPSID>	Operating system user under which the TREX processes run.
User <j2eeadm>	Operating system user that you use to log on to the host on which the J2EE Engine is running.

### Abbreviations

The following abbreviations are used in the graphics.

Abbreviation	Meaning
<b>Master servers</b>	
M NS	Master name server
M IS	Master index server
M QS	Master queue server
<b>Slave server</b>	
S NS	Slave name server
S IS	Slave index server

<b>Backup server</b>	
B IS	Backup index server
B QS	Backup queue server
<b>Other servers</b>	
RFC	RFC server
WS	Web server
PP	Preprocessor
<b>Data</b>	
MI	Master index
SI	Slave index
SN	Index snapshot
Q	Queue
T	Topology file

### Commands

Commands such as script calls are sometimes distributed over several lines in this documentation. When you execute these commands, enter them as one line.



## Required Documentation

To implement a distributed system, you require:

- The *SAP NetWeaver 7.0 Search and Classification (TREX) **Multiple Hosts*** installation guide

This installation guide details the installation and configuration of a distributed TREX system on more than one host. The installation guide is located on the *SAP Service Marketplace* at [support.sap.com/sltoolset](http://support.sap.com/sltoolset) → System Provisioning → Installation Option of Software Provisioning Manager → Manager Installation Guides - Standalone Engines and Clients → SAP NetWeaver Search and Classification TREX.

- The *SAP NetWeaver 7.0 Search and Classification (TREX) **Single Host*** installation guide

This installation guide details the installation of a TREX instance on a single host. The installation guide is located on the *SAP Service Marketplace* at [support.sap.com/sltoolset](http://support.sap.com/sltoolset) → System Provisioning → Installation Option of Software Provisioning Manager → Installation Guides - Standalone Engines and Clients → SAP NetWeaver Search and Classification TREX .

Use the *NetWeaver 7.0 Search and Classification (TREX) **Multiple Hosts*** installation guide as a central entry point. This guide contains information relevant for planning and provides an overview of the implementation process. At appropriate points, it refers to the *SAP NetWeaver 7.0 Search and Classification (TREX) **Single Host*** installation guide.



## Fundamentals

The following sections provide an overview of the TREX architecture and introduce the concepts of distributed TREX systems.



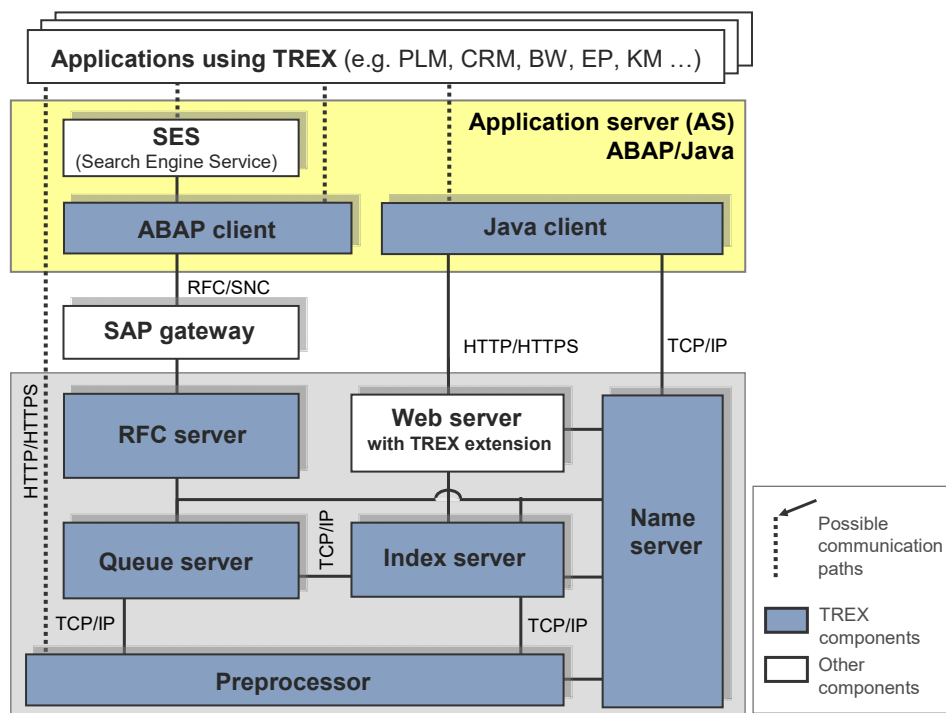
## TREX Architecture

TREX is based on a client/server architecture. The client component is integrated into the application that uses the TREX functions, and allows communication with the TREX servers. The server component processes the requests; it indexes and classifies documents and answers search queries.

The client component is subdivided into the [Java client and ABAP client \[Page 5\]](#). The server component is subdivided into the following servers:

- [Web server with TREX extension \[Page 5\]](#)
- [RFC server \[Page 5\]](#)
- [Queue server \[Page 6\]](#)
- [Preprocessor \[Page 6\]](#)
- [Index server \[Page 7\]](#)
- [Name server \[Page 5\]](#)

The graphic below shows the individual components and the communication between components:





## Java Client and ABAP Client

TREX provides programming interfaces (Application Programming Interfaces, APIs) for the languages Java and ABAP. These interfaces are also called the Java client and the ABAP client.

The interfaces allow access to all TREX functions. You can use the interfaces to create indexes and queues, to perform indexing, and to perform searches. In addition, the interfaces provide functions to query the internal status of TREX.

The interfaces are part of the NetWeaver Application Servers (NW AS).



## Web Server with TREX Extension

The Web server is responsible for the communication between Java applications and the TREX servers. The application sends requests to the Web server in XML format using HTTP/HTTPS. The Web server converts the requests to a TREX-internal format and then forwards them to the responsible TREX servers.

A TREX component that enhances the Web server with TREX-specific functions is installed on the Web server. Technically, this component is implemented as follows:

- On Windows, as an ISAPI server extension for the Microsoft Internet Information Server
- On UNIX, as a shared library for the Apache Web server



## RFC Server

The RFC server is responsible for the communication between an SAP system and the TREX servers.

The SAP system sends requests to an RFC server using an SAP Gateway. The RFC server converts the requests to a TREX-internal format and then forwards them to the responsible TREX servers.



## Name Server

The name server manages information on the entire TREX system. It makes sure that the TREX servers can communicate with each other and that they receive all necessary information. The name server has the following tasks:

- Managing topology data

The topology data includes information on the central components of a TREX system (TREX servers, indexes, and queues).

- Coordinating replication services

The replication services are only relevant for a distributed TREX system. The name server has information on which TREX server has a particular data status. It makes sure that changed data is replicated.

- Load-balancing

The name server accepts requests and distributes them to the responsible TREX servers. It is responsible for distributing indexes and search queries.

- Ensuring high availability

The name server launches several watch dogs. They constantly monitor whether the TREX servers are available. If a TREX is not available, the name server ensures that the TREX server that is down does not receive any requests.



## Queue Server

The queue server coordinates the processing steps that take place during indexing. It collects incoming document, triggers preprocessing by the preprocessor, and further processing by the index server.

The queue server enables documents to be indexed asynchronously. This has the advantage that you can control the time of indexing. For example, you can schedule indexing for times when the system load is lower because there are fewer search queries.

In addition, the queue server can trigger index replication and integration of the delta index in the main index.



## Preprocessor

The preprocessor preprocesses documents and search queries.

Document preprocessing comprises the following steps:

- Loading documents

If the application transmits the documents as URIs rather than directly, TREX resolves the URIs. This involves fetching the documents from the repository that the URIs reference.

- Filtering documents

Documents can exist in various formats, such as Microsoft Word, Microsoft PowerPoint, PDF, and so on. The preprocessor extracts textual content from the documents and then converts it into the UTF-8 Unicode format for further processing.

- Analyzing documents linguistically

Linguistic analysis involves splitting text into individual words and reducing words to base forms (stems). The preprocessor uses a lexicon that exists in several languages for this.

During search queries, the preprocessor performs a linguistic analysis. It transmits the results of the analysis to the index server, which continues the processing of the document.



## Index Server

The index server indexes and classifies documents and answers search queries. The processing takes place in the engines that belong to the index server. There are the following engines:

- Search engine:  
This engine is responsible for standard search functions such as the exact, error-tolerant, linguistic, Boolean, and phrase searches.
- Text-mining engine  
This engine is responsible for classification, searching for similar documents ('See Also' search), the extraction of key words, and so on.
- Attribute engine  
This engine is responsible for searching for document attributes such as author, creation date, and change date.



## TREX Instances and the TREX System

A TREX instance is an administrative unit that comprises the TREX server components. A TREX instance is started and stopped as a unit.

The following components belong to a TREX instance.

- A name server
- A queue server
- One or more index servers
- One or more preprocessors
- Optionally, one or more RFC servers
- Optionally, one or more Web servers

A TREX instance runs on a host. It is possible for several TREX instances to run on the same host. A TREX instance is identified by a two-character instance number. This instance number must be unique on a host.

A TREX system consists of one or more TREX instances. If it consists of only one TREX instance, it is called a single host system. If it consists of multiple connected TREX instances, it is called a distributed system.



## Distributed TREX Systems

The sections below explain concepts that are relevant for distributed TREX systems.



## Server Tasks

In a distributed system, there are multiple instances of the individual TREX servers (name server, index server, queue server, and so on).

The servers in a distributed system do not have the same rights and have different tasks. The following sections describe these tasks.



## Master, Slave, and Backup Index Servers

The index servers in a distributed system have one of the following roles:

- Master index server
- Slave index servers
- Backup index server

A master index server is responsible for indexing. In the default configuration, it is not responsible for searching.

A slave index server is responsible only for searching and not for indexing.

The separation of the master index server and slave index servers is beneficial to performance. The indexing functions are separate from the searching functions, so that there is no loss of performance during indexing runs.

A backup index server can replace a master index server if it becomes unavailable. The backup index server is inactive if the master index server is available. When the master index server restarts after becoming unavailable, it takes over its tasks from the backup server again.

You implement backup index servers in order to make indexing highly available. The indexes must be stored centrally, so that both the master and the backup index servers can have write-access to them.

The index servers are the central components of a TREX system. In principle, their role determines the load that a host has to carry. The documentation below therefore refers to the hosts according to the role of the index server: A master, slave, or backup host is a host on which a master, slave, or backup index server is running.



## Master and Backup Queue Servers

The queue servers in a distributed system have one of the following roles:

- Master queue server
- Backup queue server

The master queue server is the primary server for managing the queues.

A backup queue server can replace a master queue server if it becomes unavailable. The backup queue server is inactive if the master queue server is available. When the master queue server restarts after becoming unavailable, it takes over its tasks from the backup queue server again.

You implement backup queue servers in order to make indexing highly available. The queues must be stored centrally, so that both the master and the backup queue servers can have write-access to them.



## Master and Slave Name Servers

The name servers in a distributed system have one of the following roles:

- Master name servers
- Slave name server

The master name servers can update the topology data for the system. The slave name servers can only read the topology data.

In a distributed system you need at least two master name servers, and cannot define more than three. The system automatically defines an active master. If the active master is unavailable, the next master name server takes over the tasks.



## Preprocessor Modes

The preprocessors can run in the following different modes:

- `search` mode

The preprocessor only preprocesses search queries. In this mode the preprocessor runs on the slave hosts by default.

- `index` mode

The preprocessor only preprocesses documents. In this mode the preprocessor runs on the master hosts by default.

- `any` mode

The preprocessor's tasks are not restricted.



The modes merely define preferences for the distribution of tasks for the preprocessors. If necessary, a preprocessor carries out all tasks regardless of its mode. For example, in certain circumstances a preprocessor that runs in index mode also processes search queries. This behavior increases the availability of the system, because in principle all preprocessors are able to carry out all tasks.



## Master and Slave Index

A master index is the original version of an index. It is managed by a master index server.

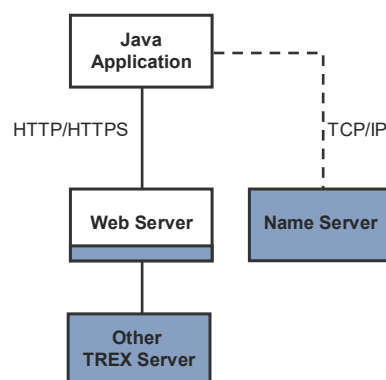
A slave index is a copy of a master index. It is managed by a slave index server. The slave index is created and updated using a replication procedure.



## Connection to the Application

### HTTP Connection

If the TREX system is connected to a Java application, the Java application communicates with both the name server and with the Web server. The Java application asks the name server via TCP/IP for the address of a Web server. It then sends the request to the Web server using HTTP/HTTPS. The Web server forwards the request TREX-internally. The graphic below depicts this communication:



There are multiple Web servers in a distributed TREX system. As soon as the Java application receives the address of one Web server, it communicates with that Web server for as long as it is available. If the Web server does not answer (for example, because it is overloaded), the Java application swaps to another Web server.

### RFC Connection

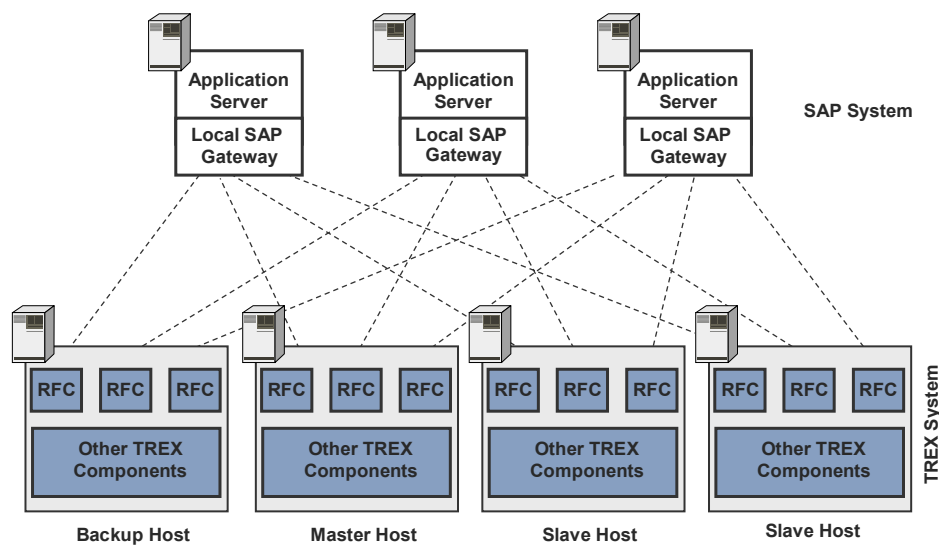
If the TREX system is connected to an ABAP application (that is, to an SAP system), both systems communicate via an RFC connection. The SAP system sends its requests to an SAP gateway. The SAP gateway sends the requests to a TREX RFC server. The TREX RFC server forwards the requests TREX-internally.

With regard to the SAP gateway, there are two variants:

- Communication takes place using the local SAP gateway of the application server.
- Communication takes place using a central SAP gateway.

In the case of a distributed TREX system, SAP strongly recommends using the local SAP gateways of the application servers. On the TREX side, TREX RFC servers are registered with each local SAP gateway. Each TREX host is connected to each application server of the SAP system.

The graphic below depicts this.



Using the local SAP gateways has the following benefits:

- The local SAP gateways process the requests quicker than a central SAP gateway.
- The SAP gateway is not a “single point of failure.” If an application server and its local SAP gateway fails, the requests are distributed among the remaining application servers and still continue to reach the TREX system.



If you use a central SAP gateway and the SAP gateway fails, the RFC connection fails too. It is not possible to switch to another central SAP gateway automatically.



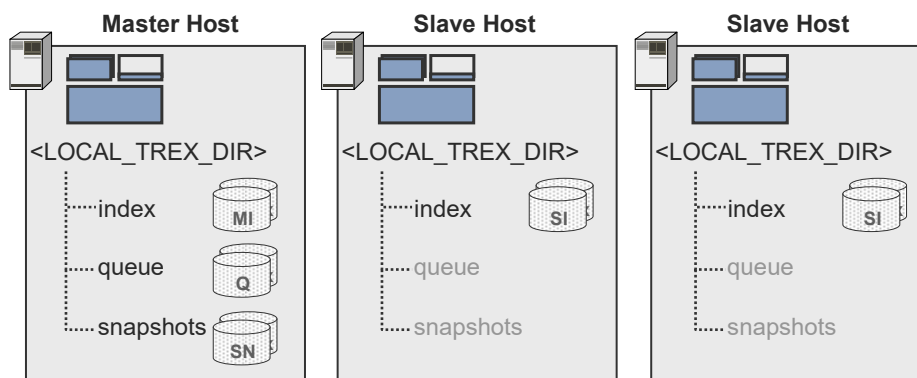
## Data Storage

In a distributed system you can keep TREX data (indexes, queues, and index snapshots) centrally or on the separate hosts.

### Decentralized Data Storage

If data is not kept centrally, each host stores its data in its own directory structure. The data is normally located locally on the hosts.

The following graphic depicts the data and directory structure with decentralized data storage:



The master indexes, corresponding queues, and the index snapshots are located on a master host. The index snapshots are index copies that the system needs for index replication.

The slave indexes are located on a slave host. They are created and updated by index replication. There is no other data on the slave hosts.

You cannot use backup hosts in systems where data storage is decentralized. This means that you cannot make indexing highly available in such systems.

### Centralized Data Storage

With centralized data storage, the data is stored so that all TREX hosts can access it.

Centralized data storage can be realized with different hardware solutions: The data can be located on a server that is optimized for file sharing, in a storage area network (SAN), or on a network attached storage server (NAS server). It is important that the connection between the TREX hosts and the data is sufficiently fast. In the following documentation, a central storage location is referred to as a file server regardless of the underlying hardware.

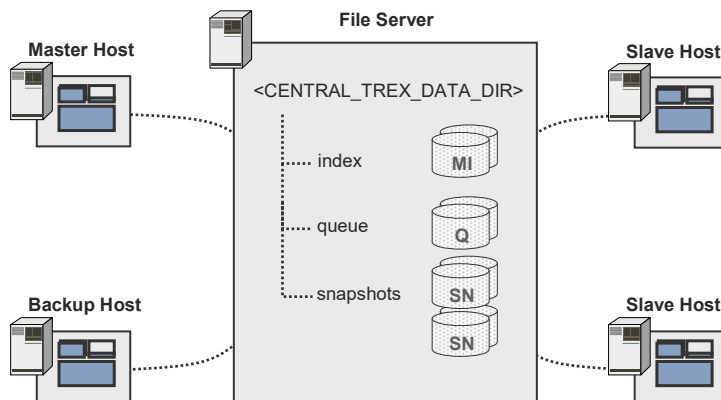
Centralized data storage is necessary if you want indexing to be highly available. You can only move from a master index or queue server to a backup index or queue server if you are using centralized data storage. You can use standard solutions such as the RAID system to make data highly available.

Centralized data storage also has the following advantages if you are only using master and slave hosts:

- Index replication generates less of a network load because the replicated files do not have to be copied onto every slave host.
- Index replication is quicker.

- Less disk space is required for the replicated indexes because all slave hosts share an index copy.

The following graphic depicts the data and directory structure with centralized data storage:



## Features of a Blade System

If you do not want to implement individual hosts you can install TREX on a blade system. TREX supports blade systems that run on UNIX.

A blade system consists of hosts in the form of server blades. A blade system has the advantage that the initial costs and running costs for maintaining the system are less than if you were using individual hosts.

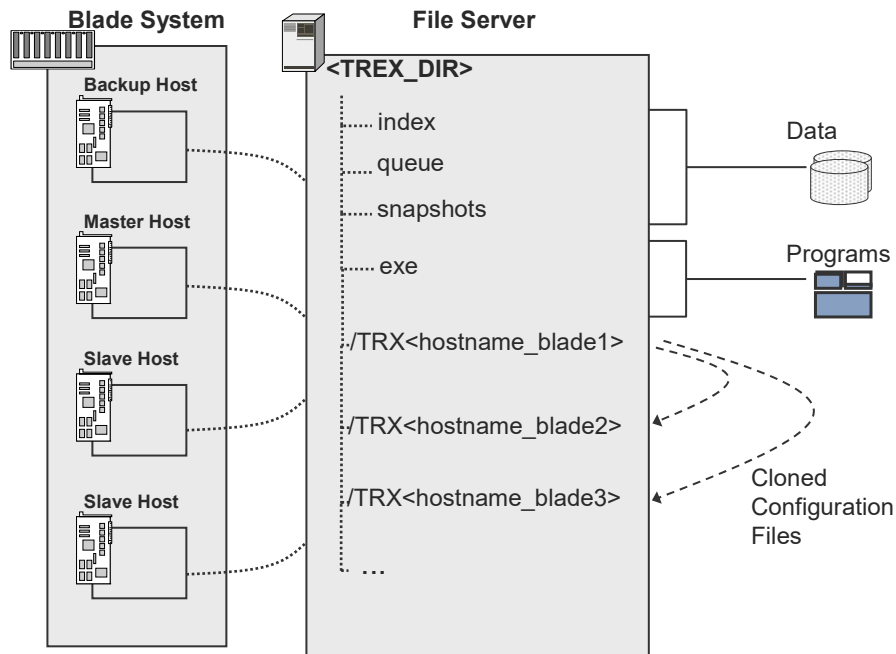
The server blades are connected to a central disk storage. This is referred to here as a file server, regardless of the underlying hardware.

The special feature of a TREX installation on a blade system is that the TREX software can be stored centrally as well as the TREX data. This means that you only have to install the software once on the file server. Maintaining the system is efficient because you only have to implement software updates once.

All server blades on which TREX is running access the same program files. However, each server blade has its own configuration files. The configuration files in the directory <TREX\_DIR> are only used as templates. A script contained in the TREX delivery creates a separate subdirectory for each server blade and copies the configuration files to this subdirectory. For more information, see [Activating the Configuration Clones for Server Blades \[Page 35\]](#).

Except for the activation of this script, the remaining configuration takes place as for a system with individual hosts.

The graphic below depicts how data, programs, and configuration files might be stored in a blade system.



## Supported Systems

There are various ways of structuring distributed systems. The table below contains an overview of the systems that are supported. Because the index servers are the central components, the systems are classed by the role of the index server.

### Supported Systems

Number and Roles of Index Servers			Data Storage	
Backup	Master	Slave	Decentralized	Centralized
–	1	1	✓	✓
–	2 or more	At least 1 per master	✓	✓
1	1	1		✓
1 for all masters	2 or more	At least 1 per master		✓
1 per master	2 or more	At least 1 per master		✓



SAP recommends configuring at least two slave index servers for each master index server, to noticeably improve the performance of the TREX search. However, all other combinations of master and slave index servers are also possible (for example, one master and three slaves). You can also start with a minimal configuration of one master index server and one slave index server.

The following is valid for all supported systems:

- You can install all systems on individual hosts or you can use a blade system.  
The graphics below depict systems on individual hosts. However, all graphics are also valid for blade systems.
- You can connect any system to an application using an HTTP connection and/or an RFC connection.  
The graphics below depict systems in which Web servers and RFC servers run. If only one type of connection is relevant, only Web servers or only RFC servers can run.

The sections below describe the supported systems in detail. In the details, the recommended ratio of one master index server to two slave index servers for improved TREX performance is assumed.



## Systems with Master and Slave Index Servers

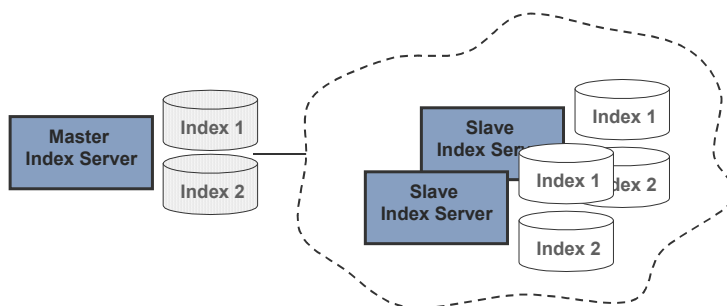
A system with master and slave index servers has the following advantages:

- Load distribution for search queries  
Parallel search queries are distributed among several slave index servers and can therefore be answered more quickly.
- High availability for searching  
Each index is available on multiple slave index servers. If one server goes down, the search queries are distributed among the remaining slave index servers. If all slave index servers becoming unavailable, the master index server would process the search queries.
- Indexing larger data sets  
A master index server can only process a certain amount of data. If you use multiple master index servers, you can index more data than in a single host system. The data must be distributed among several indexes.

If a system has no backup servers, you can store the TREX data either centrally or decentrally. The graphics below only depict systems with decentralized data storage.

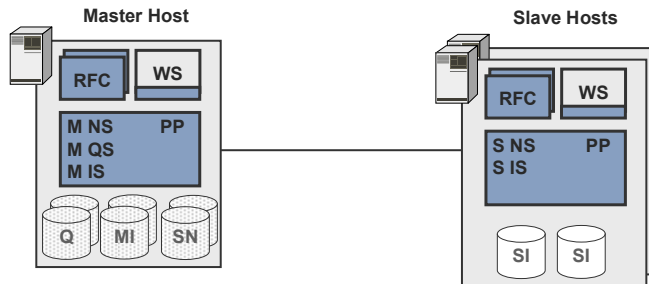
### One Master, Multiple Slave Index Servers

You can build a system with one master and several slave index servers as depicted below.



The master index server carries the entire indexing load in this scenario. The searching load is distributed among the slave index servers. Such a system is suitable for scenarios where **one** master index server can cope with the amount of data to be indexed.

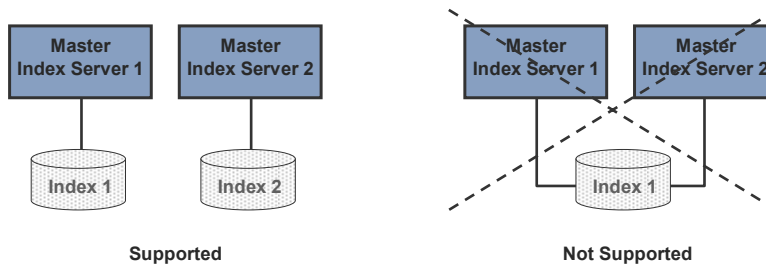
The smallest recommended system consists of one master and two slave index servers that run on separate hosts. The host that is configured as the master index server is also configured as the master queue server. The graphic below depicts the system.



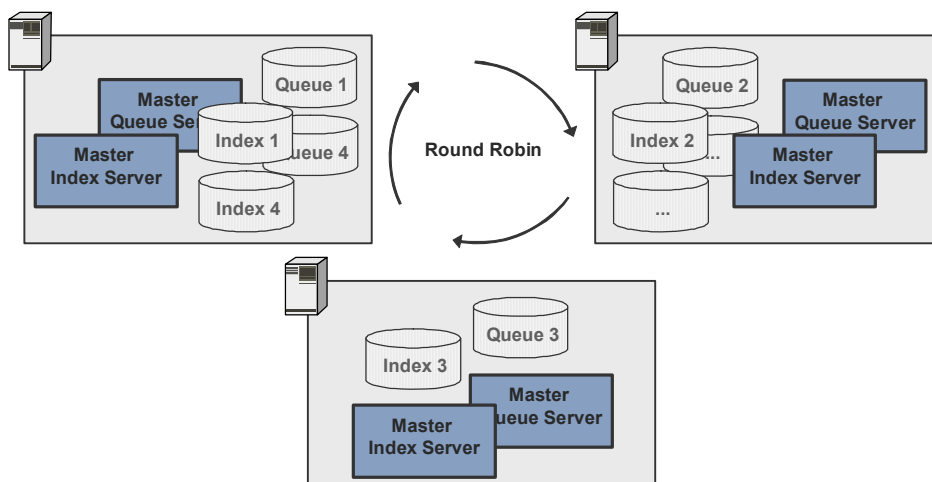
## Multiple Masters, Multiple Slave Index Servers

A master index server can only process a certain amount of data. If large data sets are to be indexed and you can distribute the data among several indexes, you can implement multiple master index servers. Each master index server manages some of the indexes.

You cannot define multiple master index servers to manage the same index.

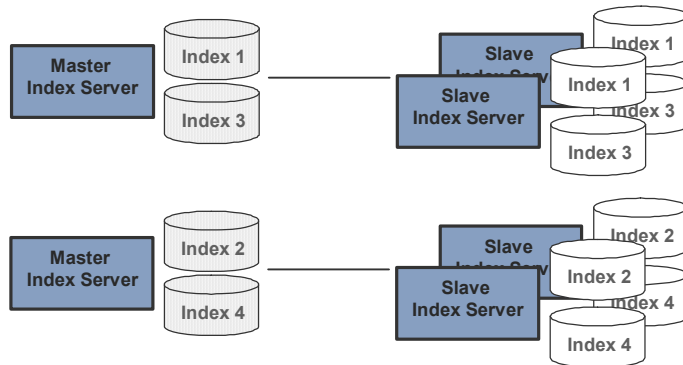


TREX distributes the indexes among the master index servers using a round robin procedure. TREX also distributes the queues among the master queue servers using a round robin procedure. Any queue is located on the same host as the master index to which it belongs.



The load on a master index server depends on how large the indexes become and how often you update the indexes. If automatic index distribution does not lead to balanced load distribution, you can change the index distribution later on.

The smallest recommended system with multiple master index servers consists of two masters, each with two slave index servers.

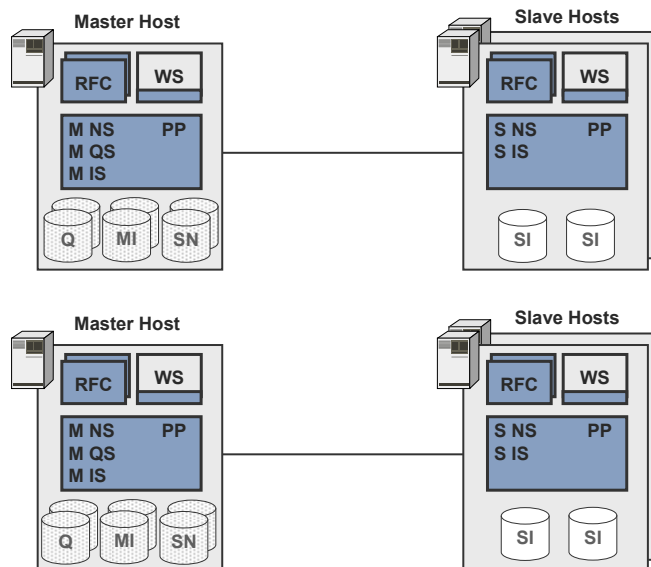


You can realize this system in two ways, according to how many CPUs and how much main memory the hosts have. For information on hardware requirements, see [Hardware, Software, and Other Requirements \[Page 25\]](#)

### One index server per host

If your hosts have few CPUs and not much main memory, only one index server can run per TREX instance. If this is the case, you distribute the master index servers among multiple hosts.

The graphic below depicts a system with two master index servers that are distributed among two hosts.



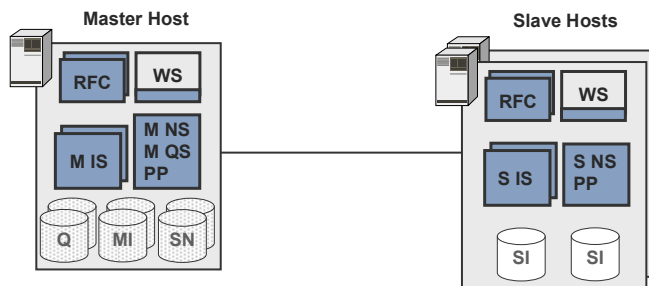
### Multiple index servers per host

If your hosts have sufficient CPUs and main memory, multiple index servers can run for each TREX instance. The TREX setup checks the hardware resources and automatically configures the number of index servers.

The same number of index servers must run on the master host and on the corresponding slave hosts. If two index servers run on the master host, two index servers must run on each slave host.

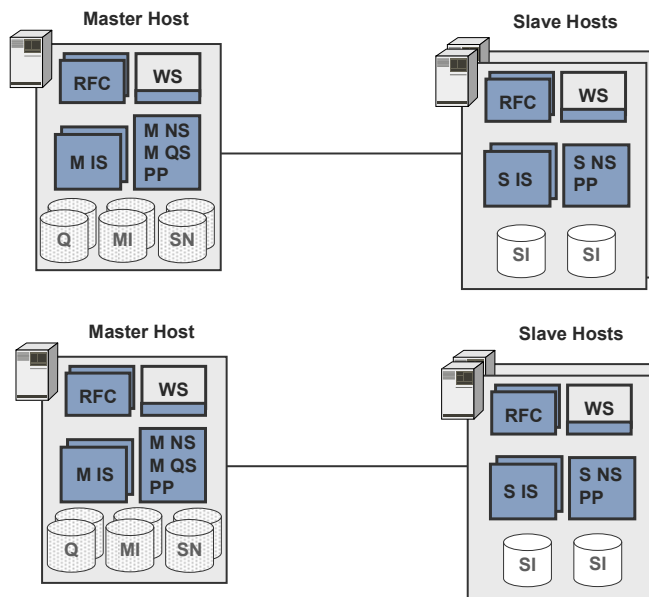


The following graphic depicts a system with two index servers for each host:



One master queue server per master host is sufficient. This server manages the queues for both master index servers running on the host.

You can build systems with multiple masters and multiple slave hosts and with multiple index servers per host. The graphic below depicts such a system.



## Systems with Master, Backup, and Slave Index Servers

You can enhance master/slave systems by adding backup servers (backup index servers and backup queue servers). Such enhanced systems offer additional high availability for indexing.

Each master index server manages some of the indexes. If a master index server goes down, indexing does not normally take place for affected indexes. You implement backup index servers to avoid this. A backup index server can replace a master index server if it becomes unavailable. The backup index server is inactive if the master index server is available.

The same is true for the queue server: If a master queue server goes down, queuing normally does not take place for the documents affected, and this means that indexing cannot take place either. You implement backup queue servers to avoid this. A backup queue server can replace a master queue server if it becomes unavailable. The backup queue server is inactive if the master queue server is available.

The TREX data has to be stored centrally in systems with backup servers. Otherwise the master and backup servers cannot access the data.

You can build systems with backup servers in the following way:

- One backup server per master server
- One backup server for all master servers

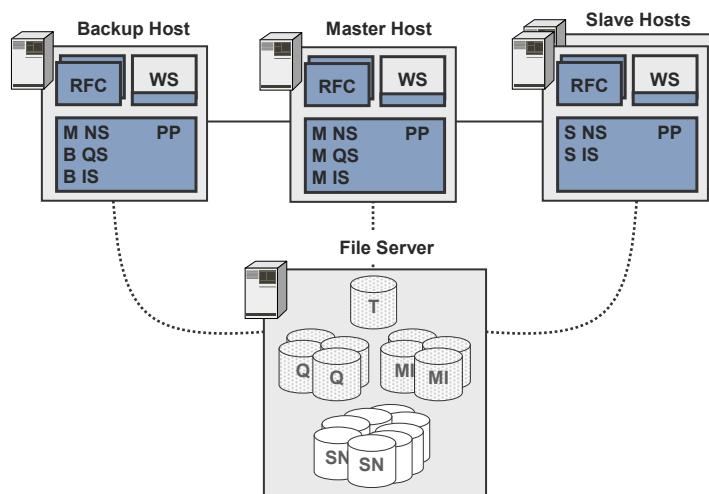
The following factors dictate which variant to choose.

- The number of master servers
- The number of master servers that are expected to be down for maintenance at the same time

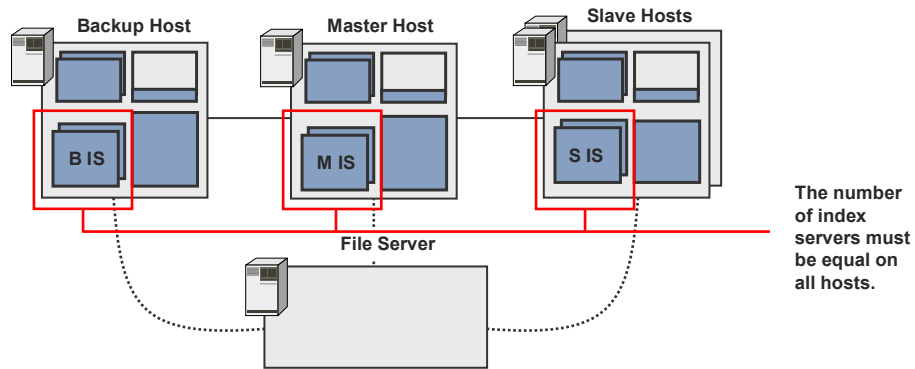
## One Backup Server per Master Server

The smallest recommended system with backup servers consists of one file server, one backup server, one master server, and two slave servers.

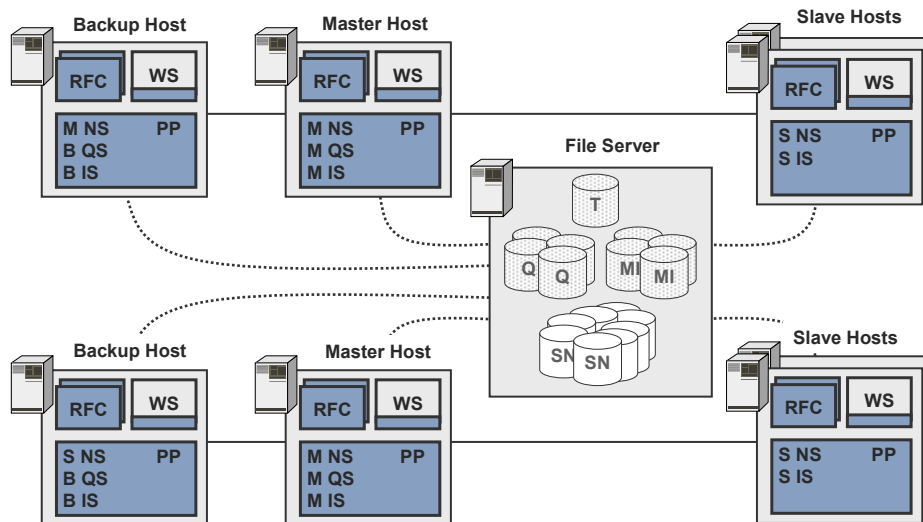
The graphic below depicts this system.



As many index servers must run on the backup host as run on the master host. If two index servers run on the master host, two index servers must run on the backup host.



The graphic below depicts a larger system with multiple master and backup hosts.

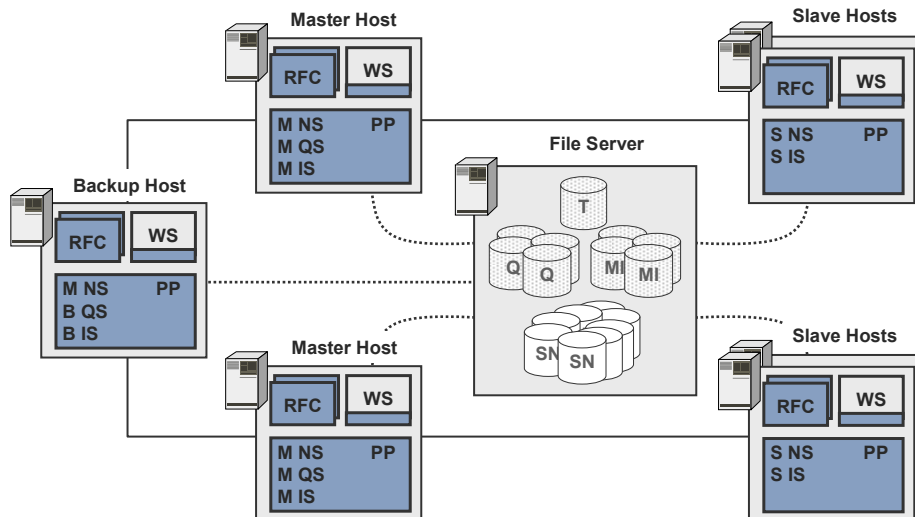


In this system, both master hosts can be down at the same time, because each has its own backup host.

## One Backup Server for All Master Servers

You can build systems in which one backup host is assigned to all master hosts. Only **one** master host can be down at any one time in such systems. If multiple master hosts with a full load go down, one backup host cannot take on the entire load.

The graphic below depicts a system with two master hosts that share a backup host.



## Summary: High Availability

If you are using TREX productively, the system has to be available for as much of the time as possible. Planned downtimes for maintenance and unplanned downtimes because of software errors should be reduced.

This section summarizes how to make searching and indexing highly available. The depicted measures are made on TREX server side and for the connection between TREX and the application. Measures that affect other software components or the hardware (highly available file servers, redundant network connection and so on) are not depicted.

## High Availability for Searching

Searching is highly available in a system with master and slave servers. If a slave host goes down, TREX forwards search queries to the other slave hosts.

It can take up to a minute to switch to another slave host. During this phase TREX search queries may not be answered. An error message may be returned.



If you have one master and two slave hosts, you can shut down **one** of the hosts for maintenance purposes (either the master or one of the slave hosts).

### Measures on TREX side

- Each master host has at least two slave hosts.
- Each index is available on at least two slave hosts.
- In systems with an HTTP connection: There are at least two Web servers.
- In systems with an RFC connection: See *RFC Connection* in [Connection to the Application \[Page 10\]](#).

### Measure on the application side

Type of Application	Measure
Java application	The Java client recognizes at least two name servers.

ABAP application	See <i>RFC Connection</i> in <a href="#">Connection to the Application [Page 10]</a> .
------------------	--

## High Availability for Indexing (Only with Queue Server)

If indexing takes place using queue servers, you can make indexing highly available. High availability means the following:

- The application can send indexing requests to TREX.
- The system automatically switches to a backup index or queue server if a master index or queue server goes down (failover). Failover is not possible in the following cases:
  - If there are network problems
  - If a file server goes down
  - If there are communication problems (application sends a request and receives no answer)

The switch to a backup index or queue server takes between 15 seconds and one minute. During this phase the system stores indexing requests in a cache and sends them to the backup server after the switch.

### Measures on TREX side

- There are at least two master name servers.
- Each master index server has a backup index server (its own or one that it shares with the other master index servers).
- Each master queue server has a backup queue server (its own or one that it shares with the other master queue servers).
- If the integration of the delta index takes place using the Python scheduler: The Python scheduler is running on all hosts that are configured as master name servers.
- If index replication takes place using the Python scheduler: The Python scheduler is running on all hosts that are configured as master name servers.
- In systems with an HTTP connection: There are at least two Web servers.
- In systems with an RFC connection: See *RFC Connection* in [Connection to the Application \[Page 10\]](#).

### Measure on the application side

Type of Application	Measure
Java application	The Java client recognizes at least two name servers.
ABAP application	See <i>RFC Connection</i> in <a href="#">Connection to the Application [Page 10]</a> .



## Global File System and TREX Instances

The TREX server software comprises two parts:

- TREX Instances

These are the program files, configuration files, and so on.

- Global TREX file system

This is a directory structure, in which information about the TREX system instances is stored. For example, this information is required by management tools to start the TREX system.

There is exactly one global TREX file system for a TREX system. When a TREX instance starts, it must have access to the global TREX file system. Otherwise, it cannot start.

When planning a distributed TREX system, you must decide which host the global TREX file system should be located on. This determines the installation steps that you work through later and which installation option you choose respectively. The following installation options are available:

Installation Option	Description
Central TREX instance	Installing a combination of a TREX instance and a global TREX file system
TREX dialog instance	Install the TREX instance only
Global TREX file system	Installing the global TREX file system only

The global TREX file system can be located on any host, as long as all TREX instances have access to it when they start. It can be located on a host that belongs to the TREX system, but it does not have to be.



SAP recommends the following for a **distributed TREX system with centralized data storage**:

Place the global TREX file system on the file server that the TREX data (indexes, queues, snapshots) is also stored on.

In order to serve as data storage, the file server and the connection between the TREX hosts and the file server must be highly available. If the global TREX file system is also located on the file server, you can be sure that the TREX instances can access it at all times.

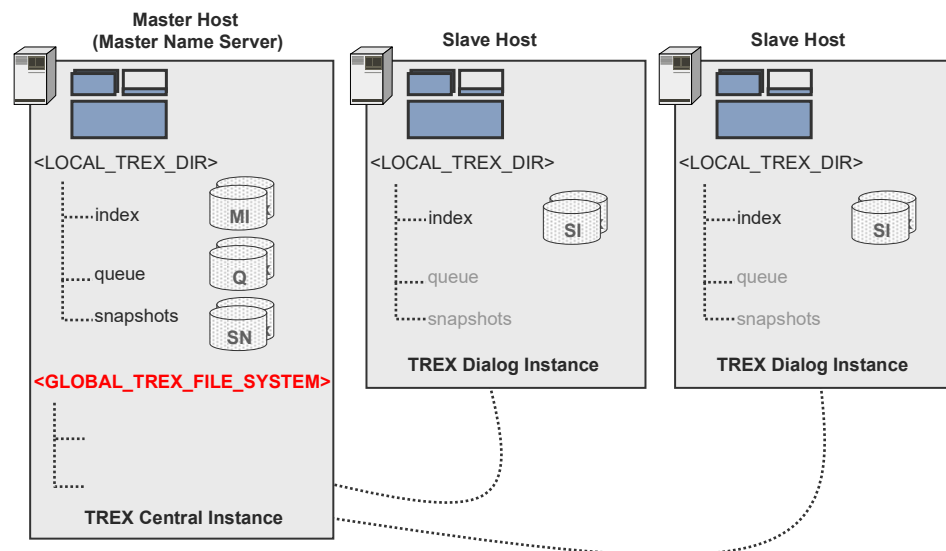
For the installation process this means that you install the global TREX file system on the file server. You install a TREX dialog instance on every master, backup, and slave host.

SAP recommends the following for a **distributed TREX system with decentralized data storage**:

Place the global TREX file system on a host that is used as the master name server.

For the installation process this means that you install a central TREX instance on the host that is used as the master name server. On all other hosts, you install TREX dialog instances.

The graphic below depicts such a scenario.






## Hardware, Software, and Other Requirements

This section lists the requirements that are unique to distributed systems. These hardware requirements relate to production systems.

### CPU, RAM, Network

Requirement Type	Requirement
CPU	<p>For <b>one</b> index server per TREX instance:</p> <ul style="list-style-type: none"> <li>At least 2 CPUs</li> <li>Recommended: 4 CPUs</li> </ul> <p>With <b>two</b> index servers per TREX instance: At least 4 CPUs. The supported processors are listed in the TREX installation guide.</p>
RAM	At least 2 GB per CPU
Network connection	At least 100 Mbit
With centralized data storage: Connection to the file server.	<p>At least 1 Gbit</p> <p> SAP recommends that you define a separate sub network.</p>



All TREX hosts must be identical as regards the number of CPUs, RAM, and network connection.

### Decentralized Data Storage: Disk Space for TREX Data



The formulas specified here are approximate and do not return exact values.

#### Required disk space on one master host

Only HTML/Text Documents	Mixed Documents (DOC, PDF, and so on)
Index size + queue (permanent) = Document set size x 2	Index size + queue (permanent) = Document set size x 0.5
Index snapshot size (permanent) = Document set size – 2 x 0.7	Index snapshot size (permanent) = Document set size x 0.5 x 0.7
Temporary disk space = Document set size x 1.5	Temporary disk space = Document set size x 0.5



We strongly recommend that you place the master indexes and the index snapshots on different hard disks. This improves performance when indexing and replicating indexes.



**Required disk space per slave host**

<b>Only HTML/Text Documents</b>	<b>Mixed Documents (DOC, PDF, and so on)</b>
Index size (permanent) = Document set size x 2	Index size (permanent) = Document set size x 0.5
Index snapshot size (temporary) = Document set size x 2 x 0.7	Index snapshot size (temporary) = Document set size x 0.5 x 0.7



The hard disk capacity and performance must be identical on master and slave hosts.



You have a document set size of 50 GB of HTML/text documents or 50 GB of mixed documents.

The following table presents the required space on the master host.

<b>Master Host</b>	<b>50 GB HTML/Text Documents</b>	<b>50 GB Mixed Documents</b>
Index + queue (permanent)	100 GB (50 GB x 2)	25 GB (50 GB x 0.5)
Index snapshot (permanent)	70 GB (50 GB x 2 x 0.7)	17.5 GB (50 GB x 0.5 x 0.7)
Temporary	75 GB (50 GB x 1.5)	25 GB (50 GB x 0.5)
<b>Total</b>	<b>245 GB</b>	<b>67.5 GB</b>

The following table presents the required space on each slave host.

<b>Slave Host</b>	<b>50 GB HTML/Text Documents</b>	<b>50 GB Mixed Documents</b>
Index (permanent)	100 GB (50 GB x 2)	25 GB (50 GB x 0.5)
Index snapshot (temporary)	70 GB (50 GB x 2 x 0.7)	17.5 GB (50 GB x 0.5 x 0.7)
<b>Total</b>	<b>170 GB</b>	<b>42.5 GB</b>

## Centralized Data Storage: Disk Space for TREX Data



The formulas specified here are approximate and do not return exact values.

### Required disk space on the file server

Only HTML/Text Documents	Mixed Documents (DOC, PDF, and so on)
Index size + queue (permanent) = Document set size x 2	Index size + queue (permanent) = Document set size x 0.5
Index snapshots size (permanent) = Document set size x 2 x 1.4	Index snapshots size (permanent) = Document set size x 0.5 x 1.4
Temporary disk space = Document set size x 1.5	Temporary disk space = Document set size x 0.5



You do not need additional disk space for the slave index. The slave index servers use one of the index snapshots as their slave index.



You have a document set size of 50 GB of HTML/text documents or 50 GB of mixed documents.

This results in the following disk requirements on the file server:

File server	50 GB HTML/Text Documents	50 GB Mixed Documents
Index + queue (permanent)	100 GB (50 GB x 2)	25 GB (50 GB x 0.5)
Index snapshots (permanent)	140 GB (50 GB x 2 x 1.4)	35 GB (50 GB x 0.5 x 1.4)
Temporary	75 GB (50 GB x 1.5)	25 GB (50 GB x 0.5)
<b>Total</b>	<b>315 GB</b>	<b>85 GB</b>

## Disk Space for TREX Software and Software Provisioning Manager 1.0

As for a single host system (see the TREX installation guide).

### Software Requirements

Requirement Type	Requirement
Operating system platform	<p>All TREX hosts must run on the same operating system platform. Mixed installations (for example, one TREX host on HP-UX and another on Windows) are not supported.</p> <p>There is no dependency between TREX and the application using TREX with regard to the operating system used. You can install TREX on a different operating system to the application that accesses TREX.</p>
TREX release	All TREX hosts must have the same TREX release with the same patch level.

The software requirements in the TREX installation guide are also valid.

### Operating System User and Permissions

The installation automatically creates the operating system user `SAPService<SAPSID>`.

In the case of a TREX system with centralized data storage, you must ensure that the user `SAPService<SAPSID>` has full access permission for the TREX data directory on the file server. Note the following:

- If the user is a network user (domain user), you have to ensure this for this one network user.
- If the user is a local user, you have to ensure this for all local `SAPService<SAPSID>` users.

In the case of a TREX system with decentralized data storage, there are no special requirements regarding access permission.

### System ID

During the TREX installation, you enter a three digit system ID, for example, `TRX`. You must use the same system ID for all TREX instances that you want to group together as a distributed system.

### TREX Instance Number

SAP recommends that you use the same instance number for all TREX instances in order to simplify administration. You define the instance number during the TREX installation.

There is only one TREX installation in a blade system with centralized program storage. The instance number is the same for all server blades. During the installation of TREX you have to choose an instance number that is still free on all the server blades on which TREX is going to run.

## TREX Daemon

You only have to change the configuration of the TREX daemon on the individual hosts under certain circumstances. These circumstances are described in this documentation.

Otherwise, you can keep the standard configuration, even if the TREX daemon starts processes that are not used. Such processes do not use up system resources and therefore do not affect performance. If you keep the standard configuration it is easy to change the roles of the hosts.



By default, a queue server runs on each host. The queue server has no function on a slave host. It is not used. You do not need to make configuration changes to the TREX daemon on the slave host.

## Connecting TREX to More Than One Application

In principle, you can connect one TREX system to more than one application. Note the following:

- The TREX system must have appropriate dimensions so that it can process the load of all the applications.
- You must take organizational measures to ensure that the applications use separate index namespaces.



## Constraints

Note the following constraints for distributed systems.

### TREX Instances

The TREX instances that form a distributed system must run on different hosts. You cannot combine several TREX instances on the same host to form a distributed system.

### Hosts

- SAP recommends using a maximum of 4 master hosts.
- SAP recommends using a maximum of 2 slave hosts per master host

For information on equipping the hosts, see [Hardware, Software, and Other Requirements \[Page 25\]](#)

### Master name servers

You need at least two master name servers, and cannot define more than three. Keep to the following rules:

- First distribute the master name servers on the master hosts.
- If there is a backup host, distribute the other master name servers there.
- If there is no backup host, distribute the other master name servers on the slave hosts.

## Indexes

- You can have a maximum of 50 master indexes per master index server.
- If the master index server has 2 GB working memory per CPU, the maximum size of a master index is 100 GB.

## Planning

### Purpose

In the planning phase you analyze your requirements and define the structure of the distributed system. An analysis of your requirements shows you how many hosts you need and the tasks that the hosts will carry out.

To simplify the installation and configuration of the system, you should create the following during the planning phase:

- A graphical depiction of the distributed system
- A table containing the host names and roles of the hosts involved

### Example

Graphical depiction of the system:

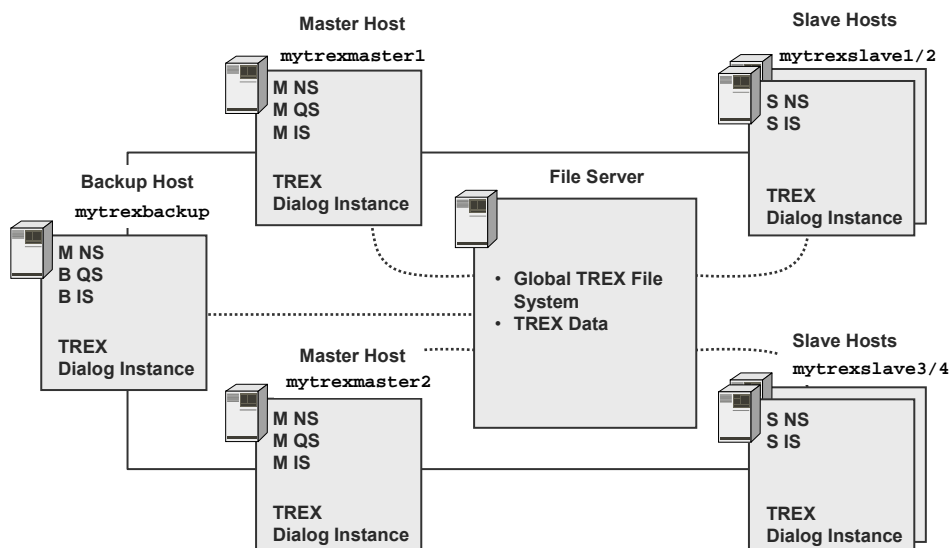


Table with system information:

Host Name	Installation Option To Use	Role	Comment
myfileserv	Global TREX file system	File server	Storage location for: <ul style="list-style-type: none"> <li>Global TREX file system</li> <li>TREX data (indexes, queues, index snapshots) and topology file</li> </ul>
mytrexmaster1	TREX dialog instance	Master name server Master index server Master queue server	Master host, manages part of the master indexes
mytrexmaster2	TREX dialog instance	Master name server Master index server Master queue server	Master host, manages part of the master indexes
mytrexbackup	TREX dialog instance	Master name server Backup index server Backup queue server	Backup host for mytrexmaster1 and mytrexmaster2
mytrexslave1	TREX dialog instance	Slave name server Slave index server	Slave host for mytrexmaster1
mytrexslave2	TREX dialog instance	Slave name server Slave index server	Slave host for mytrexmaster1
mytrexslave3	TREX dialog instance	Slave name server Slave index server	Slave host for mytrexmaster2
mytrexslave4	TREX dialog instance	Slave name server Slave index server	Slave host for mytrexmaster2



## Setting Up a Distributed System

### Purpose

If you are implementing a distributed system, you initially install all server software on each host. You then configure the hosts according to the tasks that each host is to carry out.

The following sections describe how you set up a distributed system from scratch. All tasks that are necessary for the initial configuration of the system are described.



## Checklists

### Purpose

The procedure depends on the following:

- The type of data storage you use (centralized or decentralized)
- The hardware you are using (individual hosts or blade system)

Below are the checklists for the different scenarios.

#### Individual hosts with central data storage

✓	Action
	<b>On the file server:</b>
	Install a TREX global file system (see the TREX installation guide)
	<a href="#">Create a Central TREX Data Directory [Page 33]</a>
	<b>On all TREX hosts</b>
	Install a TREX dialog instance (see the TREX installation guide)
	<a href="#">Only UNIX: Mount the Central TREX Data Directory [Page 34]</a>
	<a href="#">Only Windows: Define a network drive for the central TREX data directory [Page 34]</a>
	Start TREX
	<b>On a future master name server</b>
	<a href="#">Configure the Landscape [Page 37]</a>
	<b>With an RFC connection: On all TREX hosts and in the SAP system</b>
	<a href="#">Configure the RFC Connection [Page 51]</a>
	<b>With an HTTP connection: On the J2EE Engine</b>
	<a href="#">Configure the HTTP Connection [Page 56]</a>

#### Blade system with central program and data storage

✓	Action
	<b>On the file server:</b>
	Install TREX (see the TREX installation guide <i>Single Host</i> )
	<a href="#">Activate the Configuration Clones [Page 35]</a>
	<b>On all server blades on which TREX is to run</b>
	Start TREX
	<b>On a future master name server</b>
	<a href="#">Configure the Landscape [Page 37]</a>
	<b>With an RFC connection: On all server blades and in the SAP system</b>
	<a href="#">Configure the RFC Connection [Page 51]</a>
	<b>With an HTTP connection: On the J2EE Engine</b>
	<a href="#">Configure the HTTP Connection [Page 56]</a>

**Individual hosts with decentralized data storage**

✓	<b>Action</b>
	<b>On a future master name server</b>
	Install a central TREX instance (see the TREX installation guide)
	<b>On all other TREX hosts</b>
	Install a TREX dialog instance (see the TREX installation guide)
	<b>On all TREX hosts</b>
	Start TREX
	<b>On a future master name server</b>
	<a href="#">Configure the Landscape [Page 37]</a>
	<b>With an RFC connection: On all TREX hosts and in the SAP system</b>
	<a href="#">Configure the RFC Connection [Page 51]</a>
	<b>With an HTTP connection: On the J2EE Engine</b>
	<a href="#">Configure the HTTP Connection [Page 56]</a>

**Preparing for Centralized Data Storage****Purpose**

If you want to store the TREX data centrally on a file server, you have to prepare first. The sections below describe the steps necessary for this.

**Creating a Central TREX Data Directory****Procedure on UNIX**

1. Create a directory for the TREX data on the file server.
2. Make sure that the directory belongs to the user `SAPService<SAPSID>`.
3. Share the directory so that all TREX hosts have full permission (read, write, and execute) for it.

The exact procedure is described in the documentation for your operating system platform.

**Procedure on Windows**

1. Create a directory for the TREX data on the file server.
2. Share the directory so that the user `SAPService<SAPSID>` has full permission for it.

The exact procedure is described in the documentation for your operating system platform.





## Only UNIX: Mounting the Central TREX Data Directory

### Procedure

Use `mount` to mount the TREX data directory that you created on the file server onto all TREX hosts. Note the following:

- Mount the directory in the same place (mount point) in the file system on all TREX hosts.



You created the directory `mytrexdir` on the file server. You mount this directory on all hosts at `/mymountpoint/mytrexdir`.



This mount point must be the same on all hosts. Otherwise, the system cannot swap from a master server to a backup server. Moreover, the slave servers cannot use a common slave index.

- Make sure that the user `<sapsid>adm` has full permission (read, write, and execute) for this directory.
- Make sure that the directory will be automatically mounted if the host is rebooted before starting TREX.

The exact procedure is described in the documentation for your operating system platform.



## Only Windows: Defining the Network Drive

### Use



SAP strongly recommends that you define a network drive on all TREX hosts for the central TREX data directory on the file server. Access via a network drive is much quicker than access without a drive.

The procedure below describes the required configuration steps.

### Procedure

Edit the configuration file `TREXDaemon.ini` on all TREX hosts as follows:

```
TREXDaemon.ini
```

```
[mappings]
```

```
map_<network_drive_letter>=\\<file_server>\<TREX_data_directory>
```



Define the same network drive on all TREX hosts: Use the same network drive letter and specify the same directory.



In the standard system, the system uses the user `SAPSystem<SAPSID>` to access the network drive.

If you want the system to use a different user for access, specify this as follows:

```
map_<network_drive_letter>=\\<file_server>\<TREX_data_directory>
user_<network_drive_letter>=<user_name>
password_<network_drive_letter>=<password_in_plain_text>
```

## Result

The changes take effect when TREX is next started.

## Example

You have created the directory `mytrexdir` on the file server `myfileserv` and shared it as `mytrexshare`. You want to connect the directory on all TREX hosts as the network drive T:.

Configuration in `TREXDaemon.ini`:

```
[mappings]
map_t=\\myfileserv\mytrexshare
```



## Activating the Configuration Clones for Server Blades

### Use

You can install TREX on a blade system so that the TREX data and program files are stored only once on the file server and are used by all server blades. Every server blade on which TREX is running needs its own configuration files.

You use a Python script to duplicate the profile files and the configuration to all server blades in your TREX landscape so that each server blade receives its own configuration files.

You do this in the following steps:

## Initial Installation of TREX on a Central File Server

1. Mount the central file server.



```
/mnt/myfileserv
```



SAP recommends that you enter the directory `/mnt/myfileserv` in the configuration file `/etc/fstab`, so that the directory is automatically remounted when the host is started again.

2. Create a subdirectory, for example, <SAPSID> for the directory /mnt/myfileserver.



```
/mnt/myfileserver/<SAPSID>
```

3. Generate symbolic links (*symlinks*), which link from the directories /usr/sap/<SAPSID> and /sapmnt/<SAPSID> to the directory mnt/myfileserver/<SAPSID>.
4. Install TREX
5. Check whether TREX has been started and, if necessary, start TREX.

## Duplicating Profile Files and the Configuration to Server Blades

6. Log on with the user `root`.
7. Mount the central file server.



```
/mnt/myfileserver
```

8. Generate symbolic links (*symlinks*), which link from the directories /usr/sap/<SAPSID> and /sapmnt/<SAPSID> to the directory mnt/myfileserver/<SAPSID>.
9. Switch to the TREX directory /usr/sap/<SAPSID>/TRX<instance\_number>.
10. Set the environment variables required by TREX by executing the following Shell scripts.
  - Bourne shell `sh`, Bourne-again shell `bash`, Korn shell `ksh`:  
`. TREXSettings.sh`
  - C shell `csh`:  
`source TREXSettings.csh`
11. Execute the Python script `cloneInst.py`:  
`python exe/python_support/cloneInst.py`

## Result

The Python script `cloneInst.py` executes the following actions on the server blades that have been added:

- Create the same users on the added server blade as on the initial server blade
- Copy and modify the SAP profile files from the initial server blade
- Copy and modify the configuration files from the initial server blade
- Extend the directories `/etc/init.d` and `/usr/sap/sapservices`
- Start TREX



## Landscape Configuration

### Purpose

You use the TREX admin tool to configure the landscape. This tool has a graphical administration interface.

### Prerequisites

TREX has been started on the hosts that form the distributed system.

### Process Flow

1. [Start the TREX administration tool \[Page 100\]](#) on one of the future master name servers.
2. Go to the *Landscape Configuration* window.
3. [Define a new landscape \[Page 37\]](#).
4. [Add the remaining hosts \[Page 38\]](#).
5. [Define the roles of the hosts \[Page 38\]](#).
6. [Configure centralized data storage if required \[Page 40\]](#).
7. [Check and activate the configuration \[Page 42\]](#).

### Result

You have now defined the structure of a distributed system. You now have to configure the delta index and index replication. For more information, see [Delta Index and Index Replication Configuration \[Page 58\]](#).



## Defining a New Landscape

### Use

The local host is entered in the *Hosts* table in the *Landscape Configuration* window. By default, the local name server has been defined during the TREX installation as the *First (1st ) Master Name Server* (see the column *Name Server Mode*) and as the *Master Index Server* and the *Master Queue Server*. Since the local name server is already preconfigured as the master name server, you can use it as the starting point for configuring your distributed TREX system landscape.

### Procedure

Enter a meaningful description for the new TREX system landscape.



## Adding a Host

### Use

You use the preconfigured master name server as the starting point for the configuration of your distributed TREX system landscape and then add the remaining TREX hosts to it. Note the following:

- If multiple TREX instances are running on a host, you can only add one of them.
- You can only add TREX instances that belong to no other distributed system.
- You can only add TREX instances that have the same system ID.

### Procedure

1. Choose *Add Host*.
2. Enter the address of the name server that runs on the host to be added. The name server port is

`3<trex_instance_number>01`



If the instance number is 48, the name server port is 34801.



## Defining the Roles of Hosts

### Use

After you have added hosts to the distributed system, you define which roles the hosts are to have. There are the following roles:

- Master name servers  
There can be up to three master name servers in a distributed system. At least two must be defined. See [Constraints \[Page 29\]](#) for information on the hosts on which the master name servers must be located.
- Master index servers and master queue servers
- Slave index servers
- Backup index servers and backup queue servers

### Defining a Master Name Server

1. Select the required host in the *Hosts* table.
2. In the column *Name Server Mode*, choose *1st master*, *2nd master*, or *3rd master*.

### Defining a Master Index Server or Master Queue Server

1. Select the required host in the *Hosts* table.
2. Select *Master Index/Queue Server*.

## Defining a Slave Index Server

1. Select *Use Slave Index Servers*.
2. Define the slave index server in the *Hosts* table.
  - a. Select the host that you want to define as a slave index server.
  - b. In the column *Slave Index Server for...* specify the master index server to which the server belongs.

## Defining a Backup Index Server or Backup Queue Server

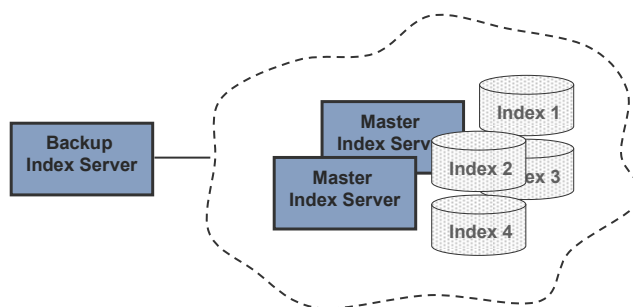
1. Select *Use Backup Index/Queue Servers*.
2. If the master servers are to share a backup server, select *Use One Shared Backup Server*.



The graphics below only depict the master and backup index servers. The graphics are also valid for master and backup queue servers.

### Example 1

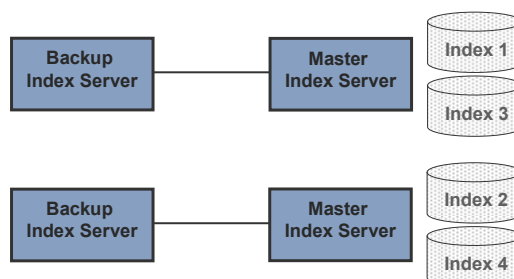
The following system has two master servers that are sharing a backup server.



In this case, select *Use One Shared Backup Server*.

### Example 2

In the following system, each master server has its own backup server.



In this case, do not select *Use One Shared Backup Server*.

3. Define the backup server in the *Hosts* table.
  - a. Select the host that you want to define as a backup server.
  - b. If you have selected *Use One Shared Backup Server*, you just need to indicate that the host is the backup server. If you did not select this field, specify the master server to which this server belongs in the column *Backup Index/Queue Server for...*



## Configuring Centralized Data Storage

### Use

If you want to store the TREX data centrally on a file server, specify this fact when configuring the landscape.



If you are using a file server, TREX automatically stores a topology file on the file server. The master name servers then share this topology and no longer use their local topology files.

This has the advantage that the master name servers do not need to synchronize their local topology files. In some circumstances, synchronization can cause a master name server to use an out-of-date topology file because it did not receive all of the changes.

Example: A host on which a master name server is running has not been in operation for some time. Its local topology file is therefore out-of-date. If you stop all TREX hosts and then start the master name server that has been out of operation first, the system will use its out-of-date topology file. If this happens, update the topology file manually using backup copies.

### Prerequisites

[You have prepared for centralized data storage \[Page 33\]](#).

### Procedure

1. Select *Use a File Server*.
2. **On all hosts**, change the path specifications so that they reference the central TREX data directory on the file server:
  - a. Select a host in the *Hosts* table.
  - b. Enter the relevant central TREX data directory on the file server (UNIX) or network drive (Windows) in the *Base Path* column.

## Examples of Path Specifications

### UNIX: Individual hosts with a file server

You have created the central TREX data directory `mytrexdir` on the file server `myfilesserver`. This directory is mounted at `/mypath/mytrexdir` on all TREX hosts. The path specifications are as follows:

Host	Base Path
mytrexhost_1	/mypath/mytrexdir
...	...
mytrexhost_n	/mypath/mytrexdir

### UNIX: Blade system with a file server

You have installed TREX in the directory `usr/sap/trex_<instance_number>` on the file server `myfilesserver`. All server blades access this directory. The TREX data should be located in the installation directory. The path specifications are as follows:

Host	Base Path
mytrexhost_1	/usr/sap/<SAPSID>/TRX<instance_number>
...	...
mytrexhost_n	/usr/sap/<SAPSID>/TRX<instance_number>

You do not have to change the default value for the base path.

### Windows: Individual hosts with a file server

You have created the central TREX data directory `mytrexdir` on the file server `myfilesserver`. The directory is connected as the network drive T:. The path specifications are as follows:

Host	Base Path
mytrexhost_1	T:
...	...
mytrexhost_n	T:





## Checking and Activating the Configuration

### Use

You can check whether the landscape configuration is consistent and complete at any time. This allows you to check the effects of the configuration changes without activating them. Activate the configuration when you have made all necessary settings.

### Procedure

- To check the configuration, choose *Check*.
- To activate the configuration, choose *Deploy*.

### Result

When you check the configuration, the output area shows the checks that are carried out. If the configuration is not consistent, the system issues a message telling you so. You can use information in this message to revise your configuration.

The system also checks the configuration when you activate it. If the configuration is consistent, the system updates the configuration files of the affected hosts and restarts the servers if necessary. If the configuration has errors, the output area displays appropriate messages and does not update the configuration files.



## Example Configurations

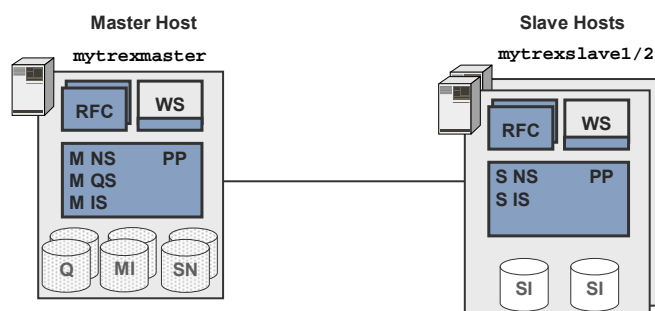
The following sections depict example systems and the relevant configurations.



## Systems with Master and Slave Index Servers

This section depicts the configuration for systems with master and slave index servers.

### One Master Index Server, Two Slave Index Servers



**Areas Scenario, Scenario Details, and Index**

Area	Field	Value
Scenario	Use Backup Index/Queue Servers	
	Use Slave Index Servers	✓
	Use a File Server	
Scenario Details	Use One Shared Backup Server	
	Assign Existing Indexes/Queues to New Backup/Slave Servers	
Index	Search on Master/Backup Server	
	Search Version	majority
	Replication Threads	1

**Hosts table (extract 1)**

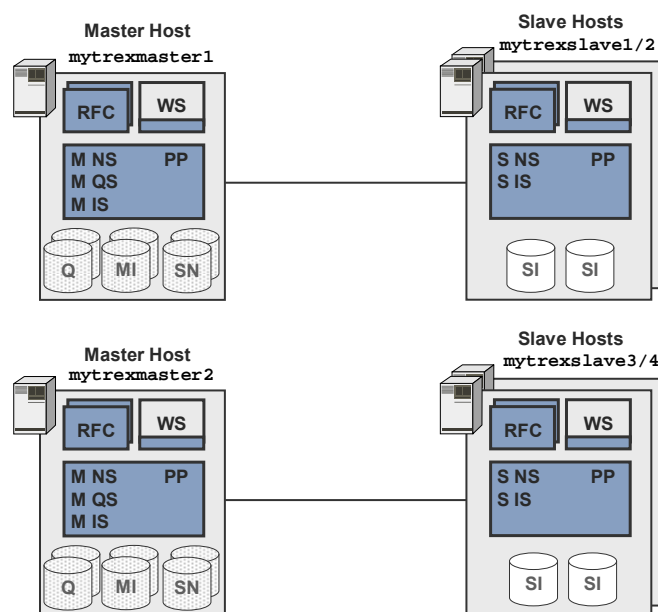
Host	Name Server Mode	Master Index/Queue Server	Slave Index Server for	Preprocessor Mode
mytrexmaster1	1st master	✓		index
mytrexslave1	slave		✓mytrexmaster1	search
mytrexslave2	2nd master		✓mytrexmaster1	search

**Hosts table (extract 2)**

Host	Base Path
mytrexmaster1	%(SAP_RETRIEVAL_PATH)
mytrexslave1	%(SAP_RETRIEVAL_PATH)
mytrexslave2	%(SAP_RETRIEVAL_PATH)

## Two Master Index Servers, Two Slave Index Servers Each

### One index server per host



TREX admin tool, *Landscape Configuration*

### Areas *Scenario, Scenario Details, and Index*

As in the previous example

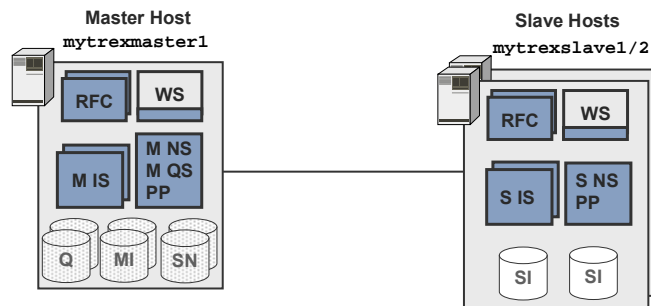
#### Hosts table (extract 1)

Host	Name Server Mode	Master Index/Queue Server	Slave Index Server for	Preprocessor Mode
mytrexmaster1	1st master	✓		index
mytrexmaster2	2nd master	✓		index
mytrexslave1	slave		✓mytrexmaster1	search
mytrexslave2	slave		✓mytrexmaster1	search
mytrexslave3	slave		✓mytrexmaster2	search
mytrexslave4	slave		✓mytrexmaster2	search

#### Hosts table (extract 2)

Host	Base Path
mytrexmaster1	%(SAP_RETRIEVAL_PATH)
mytrexmaster2	%(SAP_RETRIEVAL_PATH)
mytrexslave1	%(SAP_RETRIEVAL_PATH)
mytrexslave2	%(SAP_RETRIEVAL_PATH)
mytrexslave3	%(SAP_RETRIEVAL_PATH)
mytrexslave4	%(SAP_RETRIEVAL_PATH)

## Two index servers per host



TREX admin tool, *Landscape Configuration*

### Areas Scenario, Scenario Details, and Index

As in the previous example

#### Hosts table (extract 1)

Host	Name Server Mode	Master Index/Queue Server	Slave Index Server for	Preprocessor Mode
mytrexmaster1	1st master	✓		index
mytrexslave1	slave		✓mytrexmaster1	search
mytrexslave2	2nd master		✓mytrexmaster1	search

#### Hosts table (extract 2)

Host	Base Path	Services
mytrexmaster1	%(SAP_RETRIEVAL_PATH)	nameserver, preprocessor1, indexserver1, queueserver, indexserver2
mytrexslave1	%(SAP_RETRIEVAL_PATH)	nameserver, preprocessor1, indexserver1, queueserver, indexserver2
mytrexslave2	%(SAP_RETRIEVAL_PATH)	nameserver, preprocessor1, indexserver1, queueserver, indexserver2

#### TREXDaemon.ini on all hosts (extract)

```
[daemon]
programs=nameserver, preprocessor1, indexserver1, queueserver, indexserver2
```



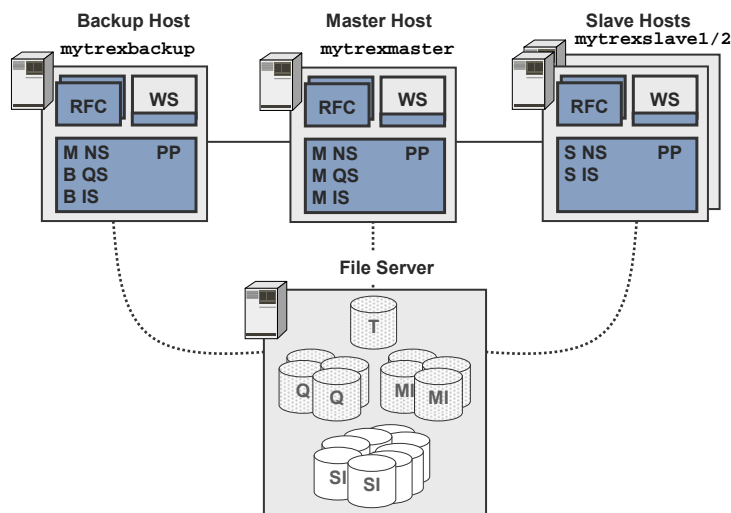
## Systems with Master, Backup, and Slave Index Servers

This section depicts the configuration for systems with master, backup, and slave index servers. The systems differ as to the number and assignment of backup index servers. The following variants are taken into account:

- One backup index server per master index server
  - One backup index server, one master index server
  - One backup index server, two master index servers
- One backup index server for all master index servers

The same specifications are valid for the master and backup queue servers.

### One Backup Index Server, One Master Server



TREX admin tool, *Landscape Configuration*

#### Areas *Scenario*, *Scenario Details*, and *Index*

Area	Field	Value
<i>Scenario</i>	<i>Use Backup Index/Queue Servers</i>	✓
	<i>Use Slave Index Servers</i>	✓
	<i>Use a File Server</i>	✓
<i>Scenario Details</i>	<i>Use One Shared Backup Server</i>	
	<i>Assign Existing Indexes/Queues to New Backup/Slave Servers</i>	
<i>Index</i>	<i>Search on Master/Backup Server</i>	
	<i>Search Version</i>	<i>majority</i>

	<i>Replication Threads</i>	1
--	----------------------------	---

**Hosts table (extract 1)**

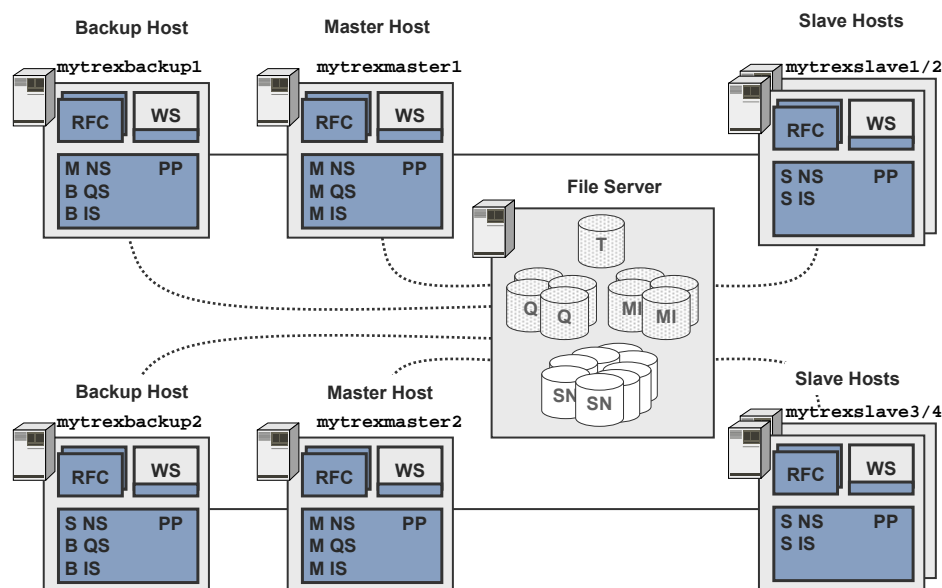
Host	Name Server Mode	Master Index/Queue Server	Backup Index/Queue Server for	Slave Index Server for
mytrexmaster	<i>1st master</i>	✓		
mytrexbackup	<i>2nd master</i>		✓mytrexmaster	
mytrexslave1	<i>slave</i>			✓mytrexmaster
mytrexslave2	<i>slave</i>			✓mytrexmaster

**Hosts table (extract 2)**

Host	Preprocessor Mode	Base Path
mytrexmaster	<i>index</i>	UNIX: /mypath/mytrexdir Windows: T:
mytrexbackup	<i>index</i>	UNIX: /mypath/mytrexdir Windows: T:
mytrexslave1	<i>search</i>	UNIX: /mypath/mytrexdir Windows: T:
mytrexslave2	<i>search</i>	UNIX: /mypath/mytrexdir Windows: T:

**Windows: TREXDaemon.ini on all hosts (extract)**

```
[mappings]
map_t=\\myfileserver\mytrexshare
```

**Two Backup Index Servers, Two Master Index Servers**

TREX admin tool, *Landscape Configuration*

**Areas Scenario, Scenario Details, and Index**

As in the previous example

**Hosts table (extract 1)**

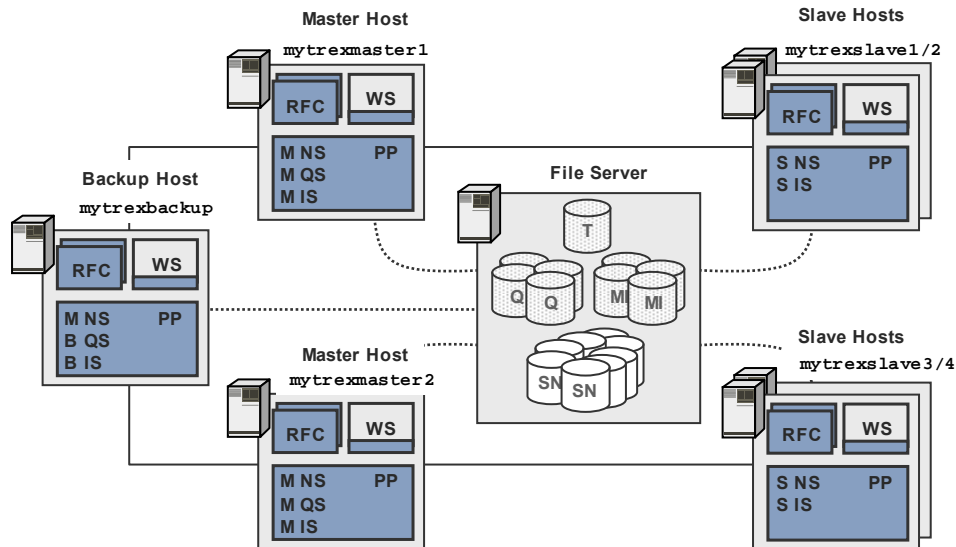
Host	Name Server Mode	Master Index/Queue Server	Backup Index/Queue Server for	Slave Index Server for
mytrexmaster1	<i>1st master</i>	✓		
mytrexmaster2	<i>2nd master</i>	✓		
mytrexbackup1	<i>3rd master</i>		✓mytrexmaster1	
mytrexbackup2	<i>slave</i>		✓mytrexmaster2	
mytrexslave1	<i>slave</i>			✓mytrexmaster1
mytrexslave2	<i>slave</i>			✓mytrexmaster1
mytrexslave3	<i>slave</i>			✓mytrexmaster2
mytrexslave4	<i>slave</i>			✓mytrexmaster2

**Hosts table (extract 2)**

Host	Preprocessor Mode	Base Path
mytrexmaster1	<i>index</i>	UNIX: /mypath/mytrexdir Windows: T:
mytrexmaster2	<i>index</i>	UNIX: /mypath/mytrexdir Windows: T:
mytrexbackup1	<i>index</i>	UNIX: /mypath/mytrexdir Windows: T:
mytrexbackup2	<i>index</i>	UNIX: /mypath/mytrexdir Windows: T:
mytrexslave1	<i>search</i>	UNIX: /mypath/mytrexdir Windows: T:
mytrexslave2	<i>search</i>	UNIX: /mypath/mytrexdir Windows: T:
mytrexslave3	<i>search</i>	UNIX: /mypath/mytrexdir Windows: T:
mytrexslave4	<i>search</i>	UNIX: /mypath/mytrexdir Windows: T:



## One Backup Index Server for All Master Index Servers



TREX admin tool, *Landscape Configuration*

### Area Scenario Details

Area	Field	Value
Scenario	Use Backup Index/Queue Servers	✓
	Use Slave Index Servers	✓
	Use a File Server	✓
Scenario Details	Use One Shared Backup Server	✓
	Assign Existing Indexes/Queues to New Backup/Slave Servers	
Index	Search on Master/Backup Server	
	Search Version	majority
	Replication Threads	1

### Hosts table (extract 1)

Host	Name Server Mode	Master Index/Queue Server	Backup Index/Queue Server	Slave Index Server for
mytrexmaster1	1st master	✓		
mytrexmaster2	2nd master	✓		
mytrexbackup	3rd master		✓	
mytrexslave1	slave			✓mytrexmaster1
mytrexslave2	slave			✓mytrexmaster1
mytrexslave3	slave			✓mytrexmaster2
mytrexslave4	slave			✓mytrexmaster2

**Hosts table (extract 2)**

Host	Preprocessor Mode	Base Path
mytrexmaster1	<i>index</i>	UNIX: /mypath/mytrexdir Windows: T:
mytrexmaster2	<i>index</i>	UNIX: /mypath/mytrexdir Windows: T:
mytrexbackup	<i>index</i>	UNIX: /mypath/mytrexdir Windows: T:
mytrexslave1	<i>search</i>	UNIX: /mypath/mytrexdir Windows: T:
mytrexslave2	<i>search</i>	UNIX: /mypath/mytrexdir Windows: T:
mytrexslave3	<i>search</i>	UNIX: /mypath/mytrexdir Windows: T:
mytrexslave4	<i>search</i>	UNIX: /mypath/mytrexdir Windows: T:

## Features of a Blade System

If you are using a blade system and the TREX data is located in the installation directory, the column *Base Path* has the following value:

**Hosts table (extract)**

Host	Base Path
mytrexmaster	usr/sap/<SAPSID>/TRX<instance_number>
...	...

The rest of the configuration is the same.



## Configuration of the RFC Connection

### Purpose

If you want to connect the TREX system to an SAP system, you must configure an RFC connection.

### Process Flow

1. [Define an SAP system user \[Page 52\]](#).
2. [Determine the connection data for the SAP system \[Page 53\]](#).
3. [Configure the RFC connection \[Page 54\]](#) using the TREX admin tool (stand-alone).



For more information about starting the TREX admin tool (stand-alone), see [Starting the TREX Admin Tool \[Page 100\]](#).

## Result

For more information about the RFC connection and handling connection and configuration errors, see the documentation on the TREX admin tool (stand-alone). You can find this documentation in the SAP Library at [help.sap.com/nw2004s](http://help.sap.com/nw2004s) → *SAP NetWeaver*.

### See also:

[Connection to the Application \[Page 10\]](#)



## Creating a SAP System User for the TREX Admin Tool (Stand-Alone)

### Use

You must create an SAP user that the TREX admin tool (stand-alone) can use to log on to the SAP system. In addition, the SAP user is required so that the TREX alert server has permission to regularly test and check the RFC configuration. When doing this, the user can have been created in the default client or in another client. In this case, make sure that you enter the associated client for the user during the [configuration of the RFC connection in the TREX admin tool \[Page 54\]](#).

The TREX admin tool (stand-alone) is used to configure and monitor TREX. You also use this admin tool to configure the RFC connection between TREX and the ABAP application that is using TREX. To use the TREX admin tool (stand-alone) to create the RFC destination, the admin tool requires an SAP system user that you create based on the predefined role `SAP_BC_TREX_ADMIN`. This user then has the authorization required to configure the RFC connection.



For more information on the `SAP_BC_TREX_ADMIN` role, see SAP Note 766516.

### Overview of the Permissions Assigned by the `SAP_BC_TREX_ADMIN` Role

Type and Scope of the Permission	Activity	Explanation
Permission check for RFC access	Execute	Name of the RFC object to be protected: <code>SYST</code> , <code>TREX_ARW_ADMINISTRATION</code>
Administration for the RFC destination	Add or generate, change, display, delete, extended maintenance	Type of entry in <code>RFCDES</code> : Start of an external program using TCP/IP
Check on the transaction code at transaction launch		Transaction code: <code>SM59</code> , <code>TREXADMIN</code> , <code>TREXADMIN_AUTH</code>

Administrating TREX	Change, display, execute	
ABAP: Program flow checks	Schedule programs for background processing, execute ABAP program, maintain variants and execute ABAP program	
ALV standard layout	Maintain	
Application log	Display, delete	

### More Information

[Configuring the RFC Connection in the TREX Admin Tool \[Page 54\]](#)

## Procedure

Create an SAP system user for the TREX admin tool (stand-alone) and assign the `SAP_BC_TREX_ADMIN` role to this user.

1. Launch transaction `SU01` (user maintenance) or choose *Administration* → *System Administration* → *User Maintenance* → *User* in the SAP menu. The *User Maintenance: Initial Screen* appears.
2. Enter a new user name and choose *Create*.
3. On the *Address* tab page, enter the personal data for the user.
4. On the *Roles* tab page, assign the `SAP_BC_TREX_ADMIN` role and thus the permission to access the SAP system to the SAP system user for the TREX admin tool (stand-alone).

## Result

This user for the TREX admin tool (stand-alone) now has the authorization required to configure the RFC connection.



## Determining the SAP System Connection Information

### Use

The TREX admin tool (stand-alone) can connect to an SAP system in two ways.

- Through a specific application server of the SAP system (variant A)
- Through the message server of the SAP system (variant B)

This variant uses the load-balancing function for the SAP system. The message server assigns the request from the TREX admin tool to any application server.

Depending on the variant used, the TREX admin tool requires different connection information for the SAP system. You must determine the connection information and specify it later in the TREX admin tool.



SAP recommends using variant B. Variant A has the disadvantage that the connection does not work if the application server is not available.

## Procedure

1. Open the SAP Logon.  
SAP Logon is the program that you use to log on to an SAP system.
2. Note the following connection information:

Connection Setup Type	Required Connection Information
Through an application server (variant A)	<ul style="list-style-type: none"> <li>• SAP system ID (SID)</li> <li>• System number</li> <li>• Application server host name</li> </ul>
Through the message server (variant B)	<ul style="list-style-type: none"> <li>• SAP system ID (SID)</li> <li>• Logon group, such as <code>PUBLIC</code></li> <li>• Message server host name</li> </ul>



## Configuring the RFC Connection in the TREX Admin Tool

### Use

You work through the steps below using the TREX admin tool (stand-alone).



Configuration of the RFC connection with the TREX admin tool (stand-alone) is only available as of SAP Basis Component SAP\_BASIS 6.20 SP58, 6.40 SP16, and 7.0 SP6. If you are using TREX with an SAP system based on an earlier support package, you have to configure the RFC connection manually as described in the SAP NetWeaver 04 Installation Guide for Search and Classification (TREX) 6.1. You can find this guide on the SAP Service Marketplace at [service.sap.com/instguides](http://service.sap.com/instguides) → *SAP NetWeaver* → *Released 04* → *Installation* → *Cross-NW* → *Installation Guide Search and Classification TREX 6.1*.

## Creating a Connection

1. In the *Landscape RFC* window, choose the *Create Connection* function.
2. Choose connection type A or B. Specify the connection data for the SAP system (see [Determining the SAP System Connection Information \[Page 53\]](#)).
3. Specify the SAP system user, the associated password, and the client that the TREX admin tool is to use to log on (see [Creating a SAP System User for the TREX Admin Tool \(Stand-Alone\) \[Page 52\]](#)).



If the SAP system user in question exists in the default client, you do not need to specify the client.


## Creating an RFC Destination

1. In the *Landscape RFC* window, choose the *RFC Destination (SM59)* function.
2. Enter the following parameters:

Field	Entry
<i>SAP System</i>	SAP system that you want to set up the connection to. The list contains all SAP systems that you have registered using <i>Create Connection</i> .
<i>RFC Destination</i>	Name of the RFC destination.
<i>Description</i>	Meaningful description of the purpose

The program ID determines under which name the TREX RFC server registers with the SAP gateway. The program ID must be unique for each SAP gateway. The TREX admin tool ensures this by generating the program ID.

3. Decide which SAP gateway you want to use. You have the following options:

Option	Comment
<i>Gateway local</i> (Default setting)	Use local SAP gateways for the application servers.
<i>Gateway central</i>	<p>Use the central SAP gateway.</p> <p></p> <p>We advise against using a central SAP gateway for distributed TREX systems. The central SAP gateway is a “single point of failure.”</p> <p>If you choose this option, enter the following additional parameters:</p> <ul style="list-style-type: none"> <li>• Host name (with domain name if necessary) or the IP address of the host on which the gateway is installed.</li> <li>• Name of the SAP gateway in the form <code>sapgw&lt;instance_number&gt;</code></li> </ul>



SAP advises against creating the RFC destination directly in the SAP system. The name of the RFC destination and the program ID must satisfy certain naming conventions. The TREX admin tool ensures that these are fulfilled.



If you nevertheless create the RFC destination directly in the SAP system, note the following:

- We recommend starting the name of the RFC destination with **TREX\_**.
- Choose the activation type *Registered Server Program*.
- Choose a program ID that is unique for the SAP gateway used.

- Use the *RFC Destinations* function to register the RFC destination in the TREX admin tool.




## Completing the RFC Configuration

1. In the *Landscape RFC* window, choose the *Connect* function.

The TREX admin tool creates the connection to all SAP systems that are known to it. Because the RFC configuration is still incomplete, the configuration status is  yellow or  red.

2. Choose *Repair All*.

The TREX admin tool completes the RFC configuration and starts the TREX RFC server.

This can take several minutes. During this time, the configuration status remains  yellow or  red. After completion of the configuration process, the status changes to  green.



Do not choose *Repair All* several times in quick succession. This would trigger the configuration process more than once and delay it.

3. Check the progress by choosing *Refresh* to update the display.



## Configuring the HTTP Connection

### Use

If you want to connect the TREX system to a Java application, you must register at least one name server with the TREX Java client.



SAP recommends that you specify all master name servers on the client side. This increases the availability of the connection between TREX and the application using it. If the Java client cannot reach one master name server, it can attempt to reach another instead.

The client-side configuration is separate from the server-side configuration. In principle, you can enter any name servers on the client side, regardless of their server-side role.

### Procedure

1. If you do not know the addresses of the master name servers, look for them in the TREX admin tool under *Landscape* → *Configuration*: *<Host>:<Name Server Port>:<Name Server Mode>*.

2. [Change the Java client parameters \[Page 101\]](#) for all server processes of the cluster as follows:

- Enter **one** master name server in the following parameter:

```
nameserver.address tcpip://<one_address>
```

- Enter the remaining master name servers – separated by spaces – in the following parameter:

```
nameserver.backupserverlist tcpip://<further_addresses>
```

Your network environment dictates whether you enter only the host name or the host name and domain.

```
tcpip://<host_name_of_trex_host>:<name_server_port>
```

or

```
tcpip://<host_name_of_trex_host>.<domain>:<name_server_port>
```



```
nameserver.address tcpip://mytrexmaster1:34801
```

```
nameserver.backupserverlist tcpip://mytrexmaster2:34801 mytrexbackup:34801
```



The addresses of the master name server must be configured for all server processes of the cluster. Otherwise the connection between the J2EE Engine and TREX cannot be established.



## Information on Breakdown of Master Servers

If you have implemented a system with master and backup servers and a master server breaks down, the backup server becomes active automatically. When the master server becomes available again, the system swaps back to the master server.

If you create an index while the master server is unavailable, TREX proceeds as follows:

- The backup server becomes the master server for this index
- This index has no backup server

The TREX admin tool displays this in the *Index Landscape* area as follows:

Index Name	...	mytrexmaster :34803	mytrexbackup :34803
mynewindex			+master,<counter>
myoldindex1		-master,<counter>	+backup,<counter>
myoldindex2		-master,<counter>	+backup,<counter>
...			

← This index was created when the master server was down.

TREX does not change this assignment, even when the master server becomes available again. You have to correct the assignment for this index as follows:

- Start the TREX admin tool on any host in the distributed system.
- Make sure that the master and backup servers are available.



3. Go to the *Index Landscape* window.
4. In the column *<light><host\_name\_master\_index\_server>:<port>*, click on the line for the index in question. Choose *Add backup here* from the context menu.
5. Click on the same cell again and choose *Switch master/backup* from the context menu.

The index now has a backup server, and the master and backup servers are assigned to the index correctly.

Index Name	...	mytrexmaster :34803	mytrexbackup :34803
mynewindex		+master,<counter>	-backup,<counter>
myoldindex1		+master,<counter>	-backup,<counter>
myoldindex2		+master,<counter>	-backup,<counter>
...			

6. Go to the *Queue Landscape* window.
7. Carry out the same changes that you just carried out for the index, but this time for the queue.



## Delta Index and Index Replication Configuration

### Purpose

Delta indexes speed up updates of the master indexes. Index replication transfers changes made on master indexes to slave indexes.

Delta indexes and index replication are deactivated by default. The best time for activating them depends on which of the following scenarios you have:

Scenario	Procedure
Initial indexing of large data sets (more than 100,000 documents)	<ol style="list-style-type: none"> <li>1. Create indexes and carry out the initial indexing of the data. During this phase, the system only carries out indexing. It does not replicate data.</li> <li>2. Activate the delta indexes.</li> <li>3. Trigger the first index replication manually.</li> <li>4. Configure regular index replication.</li> </ol>
No initial indexing of large data sets	<ol style="list-style-type: none"> <li>1. Create indexes.</li> <li>2. Configure regular index replication.</li> <li>3. Monitor the size of the master indexes during routine operation. Activate the delta indexes when a master index reaches a certain size.</li> </ol>

The sections below contain background information on delta indexes and index replication, and describe the configuration required.



## Delta Index Configuration

### Purpose

TREX provides the option of activating delta indexes. This can speed up the update of the index.

This documentation contains:

- [General information on the delta index \[Page 59\]](#)
- Information on [activating the delta index \[Page 60\]](#)
- Information on [integrating the delta index into the main index \[Page 60\]](#)

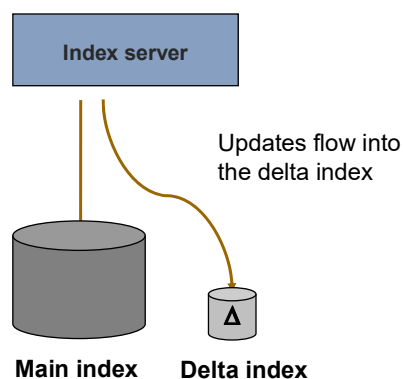


### Delta Index

When TREX updates an index, it rewrites the majority of the index files. If the indexes are large this process can take a long time and generate a high system load.

TREX allows you to activate a delta index in order to speed up the update. The delta index is a separate index that TREX creates in addition to the main index. The main index and its delta index only differ TREX-internally. Outside of TREX they form a unit.

If the delta index is activated changes flow into the delta index. Because the delta index is smaller than the main index, fewer documents are affected by the update. The delta index can therefore be updated more quickly.



The delta index is deactivated by default. The following rules are valid for its activation:

- If you have a single host system the activation is optional. However, it is recommended if the main index has reached a certain size. If you activate the delta index too soon, performance does not improve.
- If you have a distributed TREX system the activation is obligatory. However, you still only activate it once the main index has reached a certain size.

Activating the delta index doesn't only speed up the update of the master index - it also enables fast index replication with a low network load.

When index replication takes place the master index server replicates all changed master index files. Because the delta index consists of fewer files, it naturally has fewer files to replicate. This means that index replication is quicker. Moreover, if you have decentralized data storage the network load is also less because TREX has to copy less files to the slave hosts.

The delta index only speeds up the update if it is kept small. If it becomes too large, it no longer improves performance. When it reaches a certain size you have to integrate it in the main index. You can integrate the delta index manually or configure TREX so that TREX regularly integrates it automatically. TREX creates a new delta index automatically when the integration of the previous delta index is complete.

## Activating the Delta Index

### Use

The delta index is deactivated by default. You can activate it using the TREX admin tool. You activate it per index, not globally.

The best time for activating it depends on your indexing process.



SAP recommends the following:

- Initial indexing of large document sets

Activate the delta index after the initial indexing run. If you do not do this, the delta index grows too quickly and you have to integrate it into the main index earlier than you would wish. This means that you need twice the indexing time: Firstly to index the documents in the delta index, and then to integrate the delta index into the main index.

- No initial indexing of large data sets

Monitor the size of the main index during routine operation. Activate the delta index if the main index reaches 100,000 to 1,000,000 documents or 500 MB.

### Procedure

1. Go to the window *Index Admin* → *Index Info* in the TREX admin tool.
2. Select the index that you want to activate the delta index for. Choose *Delta Index On*.

## Integrating a Delta Index into the Main Index

### Use

A delta index only speeds up the update of the corresponding index if it is small. If it becomes too large, you have to integrate it into the main index. After the integration has taken place TREX creates a new delta index.

The integration process involves TREX rewriting all main index files. The duration of the integration process depends on the size of the main index. It can last a few minutes or several hours.

In a distributed system the entire main index has to be replicated after the integration has taken place. This replication takes about the same amount of time as the initial replication.

The index server cannot index new documents during the integration of the delta index. This has the following effects:

- If indexing takes place with a queue server, the queue server retains the documents until the integration process has been completed. Then the queue server transmits the documents to the index server.
- If indexing takes place without a queue server, the application can continue to send indexing requests to the index server. However, the index server only processes them after the completion of the integration process. This means that it takes longer for indexing requests to be processed and for the application to receive the relevant response.

You can trigger the integration process manually or carry it out at defined time intervals. There are two difference procedures for time-dependent integration. The procedure that you use depends on whether indexing takes place with or without a queue server (QS). The table below gives an overview of the procedures.

Procedure	Use with	
	Indexing with QS	Indexing without QS
Manual	✓	✓
Time-dependent using the queue server	✓	
Time-dependent using the Python scheduler		✓



We recommend the following for the time of the integration:

- Trigger the first integration process if the delta index is bigger than 500 MB. You can find out the size of the delta index in the window *Index Admin* → *Index Info* in the TREX admin tool.
- The integration process should take place at times when the system is not too busy.
- Do not carry out the integration process too often. With large indexes, the integration and subsequent replication of the main index takes a corresponding amount of time.

## Integrating the Delta Index Manually

1. Go to the window *Index Admin* → *Index Info* in the TREX admin tool.
2. Select the index in question and choose *Merge Delta Index*.

3.

## Integrating the Delta Index Time-Dependently Using the Queue Server

In the queue parameters enter the time for the integration in *Merge Time for Delta Index*.



Use **all (4:00)** to trigger replication every morning at 4am.



You do not need to coordinate the integration time with other activities carried out by the queue server and index server. If the activities collide, the index server coordinates when it carries out which action.

For more information on changing queue parameters, see [Configuring Queue Parameters \[Page 100\]](#).

## Integrating the Delta Index Time-Dependently Using the Python Scheduler

Change the following configuration files on **all master name servers**:

Configuration file	Change
TREXDaemon.ini	<ol style="list-style-type: none"> <li>1. Activate the Python scheduler by changing the TREX configuration file <code>TREXDaemon.ini</code> in the TREX admin tool, menu path <i>Landscape</i> → <i>Ini</i> as follows:  <pre>[daemon] programs=&lt;other_sections&gt;,cron</pre> </li> <li>2. Once you have saved the changes, the TREX admin tool asks you whether it should trigger reconfiguration so that the changes to the configuration file take effect.  Confirm this query by choosing <b>Yes</b>.</li> </ol>
crontab.ini	<p>Remove the comment sign from the following line:</p> <pre>&lt;schedule&gt; python mergeDeltaIndex.py --silent --allIndexes=1 ''</pre> <p>Modify the schedule if necessary. For information on syntax and for examples, see the configuration file.</p>



## Index Replication Configuration

### Purpose

Index replication transfers changes made on master indexes to slave indexes. The sections below describe the process and configuration of index replication.



## Index Replication Process

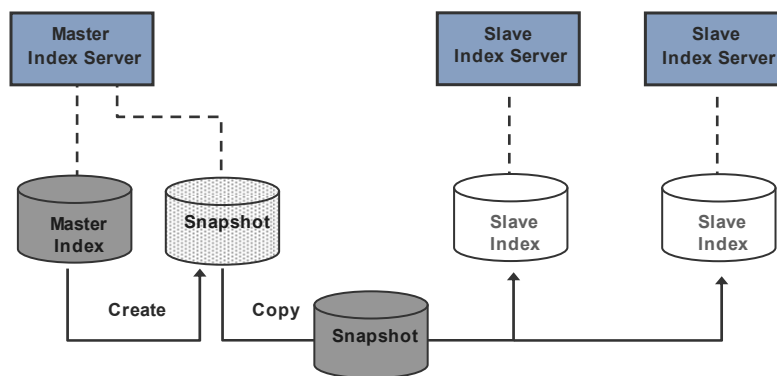
Index replication takes place in a system with master and slave index servers. The master index server manages the original indexes and the slave index servers access index copies. Replication makes sure that changes to the master indexes are transferred to the index copies.

Replication takes place in different ways depending on the type of data storage.

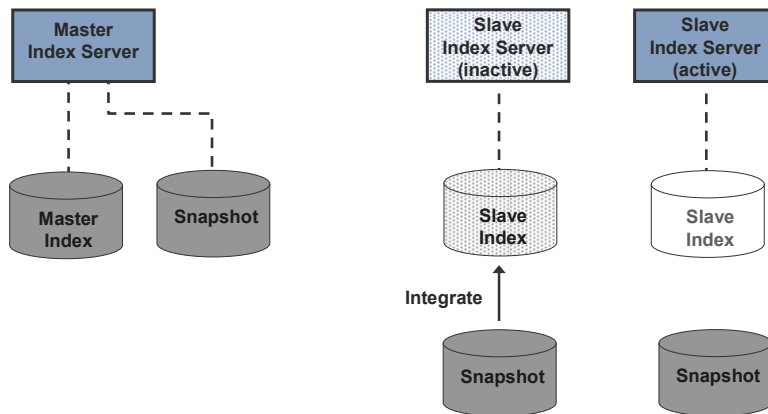
### Replication with Decentralized Data Storage

The initial replication of an index takes place as follows:

1. The master index server generates an index snapshot. The name server tells the slave index servers that the index snapshot is available. The slave index servers request the snapshot from the master index server.



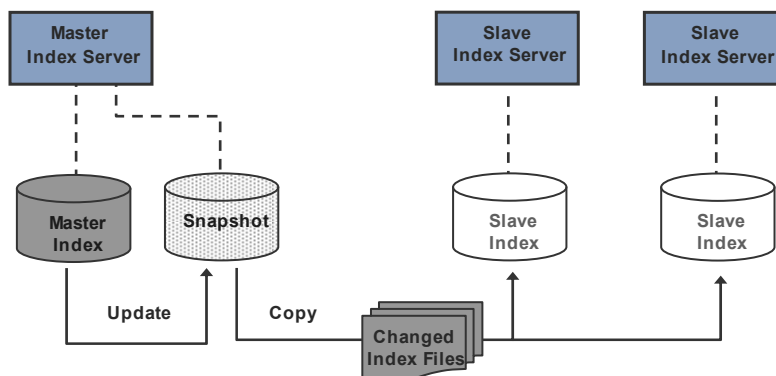
2. When all slave index servers have the index snapshot, they integrate it into their index one after the other. The slave index server currently integrating the files has the status 'inactive'. This means that it is not available for searching. It receives the status 'active' again as soon as the integration has been completed.



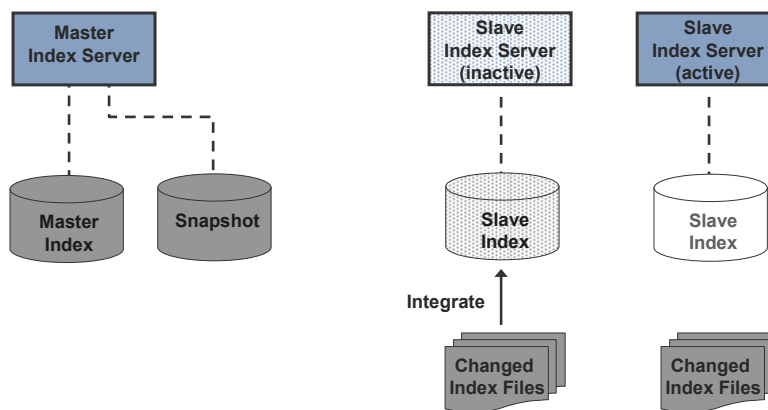
Because all index files are copied for the initial replication, the process can take a long time if the index in question is large.

In subsequent replications the system only replicates the changed index files. This is normally a smaller amount of data than for the initial replication, and subsequent replications are therefore faster. The process flow is as follows:

1. The master index server compares the master index and the index snapshot in order to determine changed index files. It then updates the index snapshot. The name server tells the slave index servers that a new index version is available. The slave index servers request the changed index files from the master index server.



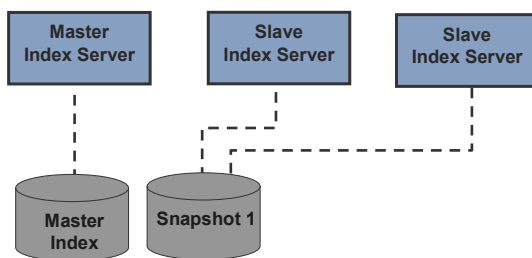
2. When all slave index servers have the changed index files, they integrate them into their index one after the other. The slave index server currently integrating the files has the status 'inactive'. This means that it is not available for searching. It receives the status 'active' again as soon as the integration has been completed.



## Replication with Centralized Data Storage

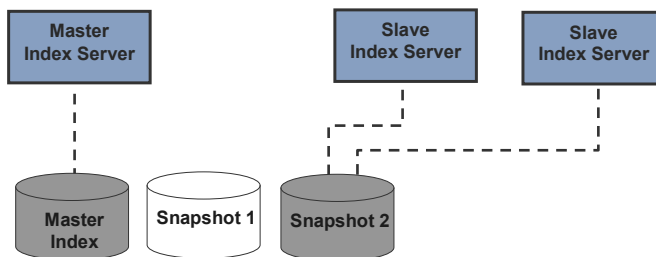
The **initial** replication of an index takes place as follows:

1. The master index server generates a complete copy of the index (index snapshot).
2. The slave index servers connect to the index snapshot and use this as their slave index.



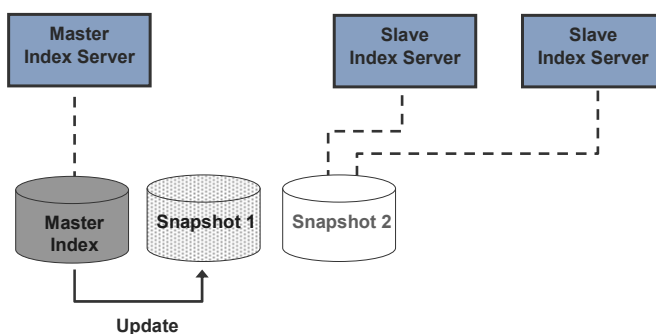
If the master index changes and replication needs to take place again, the following occurs:

1. The master index server generates a second index snapshot.
2. The slave index servers change to the second index snapshot.



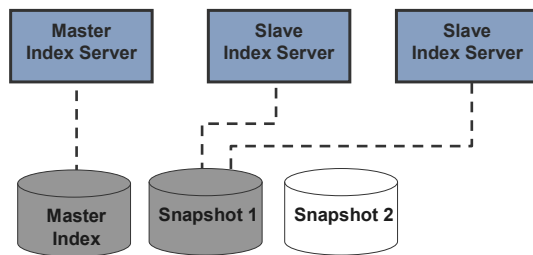
All subsequent replications take place as follows:

1. The master index server determines the changed index files by comparing the master index with the index snapshot that the slave index servers are not currently using. It then updates this index snapshot.



2. The slave index servers change to the updated index snapshot.





## Triggering Index Replication

### Use

By default index replication is deactivated. You can trigger replication in various ways. The table below gives an overview of the methods and of when you should use them.

	Procedure	Effect	Use with	
			Indexing with QS*	Indexing without QS*
1.	Manual replication	Index changes are available for searching when you have replicated the index.  You use this method for the initial replication.	✓	✓
2.	Automatic replication following optimization in the queue server ( <i>Replicate after Optimize</i> ).	Following the optimization of documents, index changes are available in the queue server for the search.	✓	
3.	Automatic replication following every index update	All index changes are available quickly for the search.	✓	✓
4.	Time-dependent replication using the queue server	Index changes are available for searching when the next replication has taken place. Replication takes place regularly according to a defined schedule.	✓	
5.	Time-dependent replication using the Python scheduler	Index changes are available for searching when the next replication has taken place. Replication takes place regularly according to a defined schedule.		✓
6.	Replication triggering by the application using TREX	TREX provides the (ABAP/Java) application with methods for triggering replication.	✓	✓

\*QS = queue server

The system replicates the entire index for the initial replication. In subsequent replications the system only replicates the changed index files. The duration of the replication and the generated system load depends on the following factors:

- Are the indexes stored centrally or are they distributed?  
With decentralized data storage the replication generates a higher net load because the system has to copy the indexes to the slave hosts.
- How often is the index updated?
- How many index files need to be replicated? This depends on the size of the index or delta index.
- How many indexes need to be replicated?
- How large are the indexes? What type of documents are indexed (documents with attributes only, documents with attributes and text content, or only documents with text content). Does the index contain text-mining information?
- How quickly should the updated information be available for searching?

In order to determine the optimum time for replication, you have to weigh up the required topicality against the system load generated.



We recommend that you carry out the initial replication manually, since it can last a lot longer than subsequent replications.



If large indexes need to be replicated frequently, it may not be possible for the system to keep to your configured interval for replication. If this is the case, the system carries out automatic replication at the next possible point in time.

## 1. Replicating an Index Manually

1. Go to the *Index Landscape* window in the TREX admin tool.
2. Carry out one of the following steps:
  - To replicate all indexes, choose *Replicate All*.
  - To replicate a single index, select the index in question and choose *Replicate Index* from its context menu.

## 2. Replicating the Index Automatically Following Optimization in the Queue Server (*Replicate after Optimize*).

Set the *Replicate After Optimize* queue parameter to *On*.

For more information on changing queue parameters, see [Configuring Queue Parameters \[Page 100\]](#).



We recommend that you arrange for the index to be replicated automatically after every update - as described in point 3 - rather than using the *Replicate After Optimize* procedure. *Replicate After Optimize* only replicates changes involving the TREX queue server. Changes made without involving the queue server, such as changes to index properties and taxonomies, are not replicated.

### 3. Replicating an Index Automatically Immediately After Every Update

When you create an index, you can arrange for it to be replicated automatically immediately after every update. To do so, proceed as follows.

1. Go to *Index* area in the *Landscape* → *Configuration* window of the TREX admin tool.
2. Activate automatic index replication by selecting the *Auto Replication* checkbox.

You can change this setting later on if you want.

3. Go to the *Index* → *Landscape* window in the TREX admin tool.
4. Use the secondary mouse button to click on the index whose index replication settings you want to change.
5. Choose *Landscape Configuration* and then *Enable Auto Replication* or *Disable Auto Replication*.



This way of triggering index replication is particularly important in scenarios that do not use a TREX queue server.

### 4. Replicating the Delta Index Time-Dependently

Enter the time at which the replication is to take place in the *Replication Time* queue parameter.



Use `A11-3` to trigger replication every three hours. Use `A11 (3:00)` to trigger replication every morning at 3am.

For more information on changing queue parameters, see [Configuring Queue Parameters \[Page 100\]](#).

### 5. Using the Python Scheduler to Schedule Index Replication

Change the following configuration files on **all master name servers**:

Configuration file	Change
TREXDaemon.ini	<p>If the Python scheduler is not yet active, activate it now:</p> <pre>[daemon] programs=&lt;other_sections&gt;,cron</pre>
crontab.ini	<p>Remove the comment sign from the following line:</p> <pre>&lt;scheduler&gt; python replicate.py --silent --allIndexes=1 ''</pre> <p>The default setting causes the system to check for changes to an index every 5 minutes. If there are no changes, the system takes no further action. If changes have taken place, they are replicated.</p> <p>Modify the schedule if necessary. For information on syntax and for examples, see the configuration file.</p>

### Result

You can monitor index replication in the TREX admin tool (stand-alone) in the *Index Landscape* window. If necessary, you can terminate replications in progress there.



## Controlling the Replication Load

### Use

You can define how many indexes the system replicates in parallel. This allows you to influence the following:

- The system load on the master index server
- If you are using decentralized data storage, the network load that arises due to copying the changed index files

The higher the number of indexes replicated in parallel, the greater the load. The lower the number, the lower the load. However, this causes replication to take longer.

### Procedure

1. Go to the window *Landscape Configuration* in the TREX admin tool.
2. In the field *Replication Threads*, enter how many indexes the system is to replicate in parallel.



## Configuring Topicality of Search Results

### Use

You can define how up-to-date you want the searched index to be. There are the following options:

Option	Meaning
<i>majority</i>	<p>The search takes place on the index version available on the majority of slave index servers. If two index versions are equally available, TREX uses the more up-to-date of the two.</p> <p>Advantage: The search queries are distributed. This setting gives the highest availability for the search because during replication TREX only switches to the new version from the old version when the majority of the slave index servers have the new version.</p> <p>Disadvantage: The search may not take place using the most up-to-date data. <i>majority</i> is the default setting.</p>
<i>latest</i>	<p>The search takes place using the most up-to-date index that has been released for replication.</p> <p>Advantage: The search takes place using the most up-to-date data.</p> <p>Disadvantage: This setting can hamper search performance. TREX always uses the up-to-date version, even if only a few (or even no) slave index servers have the most up-to-date version. If no slave index server has the most up-to-date version, the master index server receives the search queries - even if it is locked for searching. This ensures that search queries are always answered and the application receives no error message.</p>

You can change the standard configuration in the following two ways:

- For all new master indexes
- For existing master indexes

## Changing the setting for all new master indexes

1. Go to the window *Landscape Configuration* in the TREX admin tool.
2. Choose the required setting for *Search Version*.

## Changing the setting for an existing master index

1. Go to the *Index Landscape* window in the TREX admin tool.
2. Select the index in question and choose *Landscape Configuration* from the context menu.
3. Choose the required setting for *Search Version*.



## Changing a Distributed System

### Purpose

The sections below describe the changes that you can make to a distributed system after the installation. For all changes, note the [Constraints \[Page 29\]](#) that are relevant for distributed systems.



## Adding and Removing Hosts

### Features

You can use the TREX admin tool (stand-alone) to add or remove a host (server or blade server) to/from a TREX landscape. You do this if you have configured a distributed TREX landscape.

### Prerequisites

Make sure that you will still have enough CPU capacity and memory for your TREX landscape after removing a host.

### Process Flow

- [Removing a Host \[Page 71\]](#)
  - Removing a host temporarily
  - Removing a host permanently
- [Adding a Host \[Page 72\]](#)



## Removing a Host

### Use

You can use the TREX admin tool (stand-alone) to remove a host from a TREX landscape temporarily or permanently.

### Removing a Host Temporarily

1. Go to the *Landscape* → *Configuration* window in the TREX admin tool (stand-alone).
2. Remove the *Master Index/Queue Server* indicator for the host that you want to remove from your TREX landscape temporarily.
3. Choose *Check* and then *Deploy* to save your change.
4. In the *Landscape* → *Reorg* window, go to the *Plan* tab page.
5. Choose *Start Reorg* to start the required reorganization of your TREX landscape.

The reorganization process distributes indexes that are located on the removed host to other hosts. When the reorganization is finished, there are no more indexes on the host in question.



If you select the *Split/Merge Indexes* checkbox before performing the reorganization, the system not only reorganizes the indexes but also distributes and splits the logical indexes again. During this type of reorganization, the system also recalculates the number of parts of which a logical index consists.



Note that this reorganization can cause a complete reindexing process that can last as long as the initial indexing run. During this period, the system cannot perform indexing runs and searching is limited.

To add the host to your landscape again, proceed as described in [Adding a Host \[Page 72\]](#).

### Removing a Host Permanently

1. Stop TREX on the host that you want to remove from your landscape.  
The host is highlighted in red as soon as you have stopped it.
2. Go to the *Landscape* → *Configuration* window in the TREX admin tool (stand-alone).
3. Select the host that you want to remove permanently.
4. Choose *Remove Host*.

You are asked whether you want the indexes located on this host to be moved automatically.

5. Choose *Move* if you want this to happen.

The system removes all the indexes from the host in question.



After permanently removing a host, do not simply carry out an organization. For performance reasons, you should completely redistribute the indexes. To do so, select the *Split/Merge Index* checkbox in the *Landscape* → *Reorg* window of the TREX admin tool (stand-alone) and then start the reorganization. During this

type of reorganization, the system also recalculates the number of parts of which a logical index consists.



Note that this reorganization can cause a complete reindexing process that can last as long as the initial indexing run. During this period, the system cannot perform indexing runs and searching is limited.



## Adding a Host

### Use

You use the TREX admin tool (stand-alone) to add a new host (server or server blade) to your TREX landscape.

### Procedure

1. Start TREX on the host that you want to add to your TREX system landscape.
  - Install a TREX instance on the server
 

If you have not yet installed a TREX instance on the host that you want to add to your TREX landscape, do so before continuing with the procedure.

For more information about the installation of TREX, see the *SAP NetWeaver 2004s Search and Classification (TREX) Single Host* installation guide. The guide is located in the *SAP Service Marketplace* at [service.sap.com/installNW2004s](http://service.sap.com/installNW2004s).
  - Install a TREX instance on the server blade
 

For a distributed TREX installation with server blades, use the **cloneInst.py** script to generate a new TREX instance on the server blade.

See: [Activating the Configuration Clones for Server Blades \[Page 35\]](#)
2. Go to the *Landscape* → *Configuration* window in the TREX admin tool (stand-alone).
3. Add the server or server blade to your TREX landscape as follows:
  - Following the installation of an additional TREX instance on a server, execute the *Add host* command (see [Adding a Host \[Page 38\]](#))
  - The **cloneInst.py** script automatically adds the server blade to the landscape
4. Select the *Master Index/Queue Server* indicator for the host that you want to add to your TREX landscape.
5. Choose *Check* and then *Deploy* to save your change.
6. In the *Landscape* → *Reorg* window, go to the *Plan* tab page.
7. Choose *Start Reorg* to start the required reorganization of your TREX landscape.



After adding a host (server or server blade) to your TREX landscape, do not simply carry out a reorganization. For performance reasons, you should completely redistribute the indexes. To do so, select the *Split/Merge Index* checkbox in the *Landscape* → *Reorg* window of the TREX admin tool (stand-alone) and then start the reorganization. During this type of reorganization, the system also recalculates the number of parts of which a logical index consists.



Note that this reorganization can cause a complete reindexing process that can last as long as the initial indexing run. During this period, the system cannot perform indexing runs and searching is limited.



## Changing Hosts

### Purpose

You can make the following changes in a distributed system:

- Add master, backup, and slave hosts
- Remove backup and slave hosts
- Replace backup or slave hosts

For details, see the relevant sections.

### Reorganization Function in the TREX Admin Tool

You can use the reorganization function in the TREX admin tool (see *Landscape* → *Reorg*) to automatically redistribute and optimize your TREX system landscape. This function redistributes the TREX indexes among the master hosts and removes any backup and slave hosts that are not longer required (see [Optimizing the Landscape Using the Reorg Function \[Page 74\]](#)).



## Adding a TREX Index Server

### Use

You can add additional index servers to a TREX host on which an index server is already running. You do this to distribute large indexes over multiple index servers.



Each index server can use a maximum of 2GB of memory. Do not configure more index servers than can be supported by the memory on your host.

### Prerequisites

1. Open the `TREXDaemon` configuration file in a text editor.  
This file is located in the `<TREX_DIR>/<trex_host_name>` directory.
2. In the `[daemon]` section, add one or more index servers beneath the `programs` parameter: `programs=nameserver,preprocessor1,indexserver1,indexserver<next_number>,queueserver>alertserver`.  
  
Depending on the hardware of your host, one or two index servers are entered in the file by default.
3. Copy the `[indexserver1]` section and rename the copied section as `[indexserver<next_number>]`.



Repeat this procedure for each of the index servers that you want to add. Choose a new value for the port number of the additional index server (`arguments=-port <index_server_port>` parameter).

Determine the port of the first index server according to the following convention: `<index_server_port>=3<TREX_instance_number>03`. Increase the values for the port numbers in steps of ten to avoid conflicts.



If your TREX instance number is **47**:

```
[indexserver1]
arguments=-port 34703
...
[indexserver2]
arguments=-port 34713
...
[indexserver3]
arguments=-port 34723
...
[indexserver4]
arguments=-port 34733
...
```

4. Stop and start TREX so that your changes take effect.



## Optimizing the Landscape Using the Reorg Function in the TREX Admin Tool

### Use

You can use the reorganization function in the TREX admin tool (see *Landscape → Reorg*) to automatically redistribute and optimize your TREX system landscape. This function redistributes the TREX indexes on the master hosts and removes backup and slave hosts that are no longer required.



For details about using the Reorg functions in the TREX admin tool, see [Reorganization of the TREX System Landscape \[Page 75\]](#).

### Procedure

1. Check the roles of the servers in the TREX admin tool at *Landscape → Configuration*. If necessary, correct the roles and choose *Deploy*.
2. Switch to *Landscape → Reorg*:

The system automatically calculates a new, optimized distribution of your TREX system landscape according to the newly-defined roles. The new distribution of the servers is then displayed at *Landscape → Reorg → Plan* and at *Landscape → Reorg → Usage By Service*.

3. Start the reorganization of the landscape by choosing *Start Reorg in Landscape* → *Reorg* → *Summary*.

The progress of the reorganization is displayed at *Landscape* → *Reorg* → *Plan*.

## Result

The TREX system landscape has been reorganized and optimized according to your settings.



## Reorganization of the TREX System Landscape

### Use

You can use the *Reorg* function to distribute the indexes in a TREX system landscape among the available hosts to optimize their memory requirements.

The reorganization aims to achieve a balanced memory load and CPU load for the TREX system landscape.

### Integration

This function is available in the TREX admin tool (stand-alone).

You can also launch it from the *BI Accelerator Monitor*. However, several screens are used in this case.

The TREX alert server contains the *reorg* check. When this check runs, you are automatically informed by e-mail (if configured), if the system recommends a reorganization.

### Features

Based on different key figures, TREX calculate whether a reorganization should be performed. The key figures are summarized on the *Summary* tab page and displayed in detail on the *Usage by Service (I)* and *(II)* and *Usage by Index* tab pages.

The *Landscape: Reorg* window contains the following tab pages:

#### Overview of Tab Pages

Tab Page	Description
<i>Summary</i>	<p>Displays whether TREX recommends a reorganization.</p> <p>The lower table displays key figures from which TREX calculates the percentage improvement that could be achieved by a reorganization. These estimates are compared with fixed program-internal values. If the estimated improvement is high enough, TREX recommends the reorganization.</p> <p>If TREX recommends a reorganization (summary = yes), you can start the reorganization immediately or at a later time.</p> <p>To start the reorganization immediately, choose <i>Start Reorg</i>.</p> <p>To start the reorganization later, specify the date and time in the following format: YYYY-MM-DD HH:MM:SS. Then choose <i>Start Reorg</i>.</p>

<i>Plan</i>	Displays the various steps that TREX would perform during a reorganization or that are being performed during a reorganization  In addition, you see the steps and the status of the latest reorganization.
<i>Usage by Service (I)</i>	Displays the memory load and CPU load of the hosts in graphical form.  Based on these values and irrespective of the selected algorithm, TREX calculates whether a reorganization is necessary.  For example, TREX recommends a reorganization if the distribution of memory is unbalanced (identifiable by the different heights of the bar displays).  You can use a filter to show and hide the CPU load.  You cannot perform any other activities on this tab page, it is mainly for information purposes.
<i>Usage by Service (II)</i>	Displays various key figures for the hosts in table form.  You cannot perform any activities on this tab page, it is mainly for information purposes.
<i>Usage by Index</i>	Displays various key figures for all indexes in table form.  You can use a filter to define which indexes should be displayed.  You cannot perform any other activities on this tab page, it is mainly for information purposes.
<i>Interactive Reorg</i>	Displays various current key figures in table and graphical form.  On the graphic view, you can distribute the indexes manually using Drag&Drop. This function is intended for experts.
<i>Options</i>	You can define the algorithm and various parameters to be used for the reorganization.  We recommend that you use the <i>memory</i> algorithm. All other algorithms are used for test purposes.  Normally, you do not have to make any changes on this tab page.

A reorganization can be necessary in the following circumstances:

- You make changes to your TREX system landscape. For example, you remove index servers or add new ones.
- The current size of indexes does not match the initial size estimates.



If indexes are moved during the reorganization, no update of the affected indexes is possible. Indexing is interrupted for the duration of the reorganization. The affected indexes are displayed with a yellow traffic light in the *Index: Landscape* window.

## Parameter Overview

The following parameters are available on the *Options* tab page. You do not have to make any changes by default.

### Reorganization Parameters

Parameter	Description
Split Indexes	Specifies whether indexes are split into logical indexes with more than one part if the defined size is exceeded.  This specification is in KB.  This parameter is deactivated by default.
Merge Indexes	Specifies whether parts of indexes are merged if the size falls below a defined value.  This specification is in KB.  This parameter is deactivated by default.
Small Indexes	Specifies whether small indexes are distributed equally among the available hosts if the size falls below a defined value.  This specification is in KB.  This parameter is activated by default and has the size 1,000 KB.
Remove Temporary Indexes	If it is activated, temporary indexes are deleted during the reorganization.  This parameter is activated by default.

## Activities

Start the TREX admin tool (stand-alone) and navigate to the *Landscape: Reorg* → *Summary* window. If TREX recommends a reorganization (summary = yes), start the function by choosing *Start Reorg*. The display switches automatically to the *Plan* tab page. The window displays which steps are being performed. You can choose the *F5* button to update the display.

To cancel the reorganization, choose *Cancel Reorg*.

The reorganization is complete once all planned steps have been performed. The *Summary* tab page displays the status *done*.



## Adding a Master Host

### Use

You can add a new master host to a distributed system. You need to do this if the capacity of the existing master hosts is insufficient.

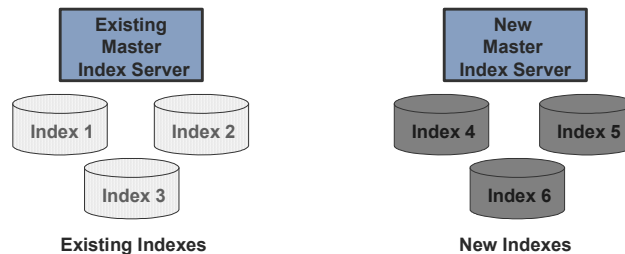
This has the following effect on the distribution of the indexes:

- The assignment of existing indexes remains unchanged.
- The new master index server receives all new indexes until all master index servers have the same number of indexes. TREX then distributes the new indexes among all master index servers according to a round robin procedure.

The same principle is used for queues.



If you previously had one master host and add another, the indexes are distributed as follows:



If you are using backup hosts, the new master host needs to receive a backup host.

- If there is one backup host for all master hosts, this backup host is automatically made backup host for the new master host.
- If each master host has its own backup host, you have to add a new backup host for the new master host.

## Procedure

1. Install TREX on the host that you want to add.
2. If you are using centralized data storage: [Mount the central TREX data directory \(UNIX\) \[Page 34\]](#) or [define it as a network drive \(Windows\) \[Page 34\]](#).
3. Start TREX on the new host.
4. Start the TREX admin tool on a host that is already configured in the distributed system.
5. Go to the *Landscape Configuration* window.
6. Use *Add Host* to add the new host.
7. Configure the new host in the *Hosts* table as follows:
  - a. Mark it as a master index/master queue server.
  - b. If you are using centralized data storage: In the column *Base Path* enter the central TREX data directory on the file server.
8. If you are using backup hosts and every master has its own backup host: [Add a new backup host \[Page 79\]](#).
9. Check the configuration. If the check does not find any errors, activate the configuration using *Deploy*.



## Adding a Backup Host

### Use

You can add a new backup host to a distributed system. You need to do this if you have added a new master host and want it to have its own backup host.

### Procedure

1. Install TREX on the host that you want to add.
2. [Mount the central TREX data directory \(UNIX\) \[Page 34\]](#) or [define it as a network drive \(Windows\) \[Page 34\]](#).
3. Start TREX on the new host.
4. Start the TREX admin tool on a host that is already configured in the distributed system.
5. Go to the *Landscape Configuration* window.
6. Use *Add Host* to add the new host.
7. Select *Assign Existing Indexes/Queues to New Slave/Backup Servers*.
8. Configure the new host in the *Hosts* table as follows:
  - a. In the *Backup Index/Queue Server for...* column, specify the master host to which the host belongs.
  - b. In the *Base Path* column, enter the central TREX data directory on the file server.
9. Check the configuration. If the check does not find any errors, activate the configuration using *Deploy*.



## Adding a Slave Host

### Procedure

1. Install TREX on the host that you want to add.
2. If you are using centralized data storage: [Mount the central TREX data directory \(UNIX\) \[Page 34\]](#) or [define it as a network drive \(Windows\) \[Page 34\]](#).
3. Start TREX on the new host.
4. Start the TREX admin tool on a host that is already configured in the distributed system.
5. Go to the *Landscape Configuration* window.
6. Use *Add Host* to add the new host.
7. Select *Assign Existing Indexes/Queues to New Slave/Backup Servers*. Otherwise the new slave host does not receive existing indexes.
8. Configure the new host in the *Hosts* table as follows:
  - a. In the *Slave Index Server for...* column, specify the master host to which the host belongs.
  - b. If you are using centralized data storage: In the column *Base Path* enter the central TREX data directory on the file server.

9. Check the configuration. If the check does not find any errors, activate the configuration using *Deploy*.



## Removing a Backup Host

### Use

You can remove a backup host from a distributed system. You may want to do this if you used the host for test purposes and no longer need it in the distributed system.

### Procedure

1. Start the TREX admin tool on any host in the distributed system.
2. Go to the *Index Landscape* window. Check the column `<light><host_name>:<index_server_port>` for the host that you want to remove.
  - a. Check which indexes the host is assigned to as the backup index server.  
If you remove this host, TREX automatically removes these assignments. The affected indexes then no longer have a backup index server. If you want indexing to be highly available, you need to assign these indexes to another backup index server.
  - b. Check whether the backup index server is currently active.  
This is displayed in the column using the entry `+backup`. You should not remove the host if the backup index server is active. If you remove the host anyway, the system does not switch automatically to using the master index server. The master index server is only assigned to the affected indexes when it is next started.
  - c. Check whether the host is assigned to any indexes as the master index server.  
If this is the case, you have to assign another master index server to the indexes before removing the host.
  - d. Check whether the host is assigned to any indexes as the slave index server.  
If you remove the host, TREX removes these assignments automatically. You have to assign these indexes to another slave index server too.

See also: [Changing Index Assignments \[Page 82\]](#)

3. Go to the *Queue Landscape* window. Check the same things for the queues as you just checked for the indexes.

See also: [Changing Queue Assignments \[Page 82\]](#)

4. If you are sure that you want to remove the host, go to the *Landscape Configuration* window.
5. Select the host in the *Hosts* table and then choose *Remove Host*.
6. Check the configuration. If the check does not find any errors, activate the configuration using *Deploy*.

### Result

The TREX instance is still installed on the removed backup host. The host may still contain configuration data with information on the distributed system. However, since these configuration files are not consistent, the TREX instance on this host will normally not start any longer. You should therefore deinstall this TREX instance.



## Removing a Slave Host

### Use

You can remove a slave host from a distributed system. You may want to do this if you used the host for test purposes and no longer need it in the distributed system.

### Procedure

1. Start the TREX admin tool on any host in the distributed system.
2. Go to the *Landscape Configuration* window.
3. Select the slave host that you want to remove in the *Hosts* table. Remove the selection in the column *Slave Index Server for*.
4. Remove the host from the landscape using *Remove Host*.
5. Check the configuration. If the check does not find any errors, activate the configuration using *Deploy*.

### Result

TREX is still installed on the removed slave host. The host may still contain index copies and configuration files with information on the distributed system. However, since these configuration files are not consistent, the TREX instance on this host will normally not start any longer. You should therefore deinstall this TREX instance.



## Replacing a Backup or Slave Host

### Use

You can replace a backup or slave host with a new host. You may want to do this if the current host needs to be maintained and will therefore be unavailable for a while.

### Replacing a Backup Host

1. [Add a new backup host to the distributed system \[Page 79\]](#).
2. [Remove the previous backup host from the distributed system \[Page 80\]](#).

### Replacing a Slave Host

1. [Remove the previous slave host from the distributed system \[Page 81\]](#).
2. [Add a new slave host to the distributed system \[Page 79\]](#).





## Changing Index Assignments

### Use

When you create an index, TREX assigns a master index server and slave index server to it. If you are using backup index servers, TREX also assigns a backup index server to the index.

You can change these assignments if necessary. You may want to do this if you need to remove a host from the distributed system and assign the indexes to other servers first.

### Prerequisites

You are using centralized data storage.

### Procedure

1. Go to the *Index Landscape* window.
2. In the table select the index whose assignment you want to change.
3. Click in a column relating to a host. Choose the required function from the context menu.

Function	Description
<i>Move master here</i>	Assigns another master index server to the index.
<i>Switch master/backup</i>	Switches the master and backup index servers. The master index server is then used as the backup index server for this index (and vice versa).
<i>Remove this backup</i>	Removes the assignment of index to backup index server.
<i>Add backup here</i>	Assigns a backup index server to the index.
<i>Remove this slave</i>	Removes the assignment of index to slave index server.
<i>Add slave here</i>	Assigns a slave index server to the index.



The functions only change the assignment of index to server. The indexes are not physically moved.

You can also change the assignments if the currently assigned master, backup, or slave index server is active at that point in time. The currently assigned server completes its current activity before the change takes effect.



## Changing Queue Assignments

### Use

When you create an index, TREX automatically creates a corresponding queue and assigns the queue to a master queue server. If you are using backup queue servers, TREX also assigns a backup queue server to the queue.

You can change these assignments if necessary. You may want to do this if you need to remove a host from the distributed system and assign the queues to other servers first.

## Prerequisites

You are using centralized data storage.

## Procedure

1. Go to the *Queue Landscape* window.
2. In the table select the queue whose assignment you want to change.
3. Click in a column relating to a host. Choose the required function from the context menu.

Function	Description
<i>Move master here</i>	Assigns another master queue server to the queue.
<i>Switch master/backup</i>	Switches the master and backup queue servers. The master queue server is then used as the backup queue server for this queue (and vice versa).
<i>Remove this backup</i>	Removes the assignment of queue to backup queue server.
<i>Add backup here</i>	Assigns another backup queue server to the queue.



The functions only change the assignment of queue to server. The queues are not physically moved.

You can also change the assignments if the currently assigned master or backup queue server is active at that point in time. The currently assigned master or backup server completes its current activity before the change takes effect.



## Allowing Searching on Master Indexes

### Use

In a distributed system, you cannot search on the master indexes by default. Search requests are answered only by slave index servers, and not by the master index servers.

The default configuration has the following advantages:

- Faster indexing

The resources on the master index server do not need to be shared among indexing and searching processes.

- Less main memory requirement

A write variant and a read variant exist for each master index. If the master index server only carries out indexing, only the write variant has to be loaded to the main memory. If the master index server carries out indexing and searching, both variants have to be loaded to the main memory.

You can change the default configuration so that the master index servers are also used for searching. This makes sense in the following cases:

- You are able to ensure that the master index servers do not index and search at the same time. This may be the case, for example, if indexing always takes place at night when there are no users using the system for searching.

- You have static indexes. These are indexes that you have created and intend to update rarely (for example, every three months).

You can change the standard configuration in the following two ways:

- For all new master indexes
- For existing master indexes

## Changing the setting for all new master indexes

1. Start the TREX admin tool on any host in the distributed system.
2. Go to the *Landscape Configuration* window.
3. Select *Search on Master/Backup Server*
4. Activate this change by choosing *Deploy*.

## Changing the setting for an existing master index

1. Start the TREX admin tool on any host in the distributed system.
2. Go to the *Index Landscape* window.
3. Select the index in question and choose *Index Properties* from the context menu.
4. Select *Search on Indexer (Master/Backup)*



## Changing Default Directories for Indexes, Snapshots, or Queues

### Use

There are default directories for indexes, snapshots, and queues. TREX creates new data in these directories. You can change the default directories. You may want to do this if you are running out of disk space in the existing default directories.

This change has no effect on existing indexes, snapshots, or queues. They remain in the previous default directories. TREX creates new data in the new default directories.



If you want to move existing indexes, snapshots, or queues, contact SAP Support.

## Procedure with Centralized Data Storage

### On UNIX

1. [Create a new TREX data directory on the file server \[Page 33\]](#).
2. [Mount the new directory \[Page 34\]](#).
3. Start the TREX admin tool on any host in the distributed system.
4. Go to the *Landscape Configuration* window.
5. Specify the new directory for all hosts in the *Basepath* column of the *Hosts* table.
6. Activate this change by choosing *Deploy*.

### On Windows

1. [Create a new TREX data directory on the file server \[Page 33\]](#).
2. Edit the configuration file `TREXDaemon.ini` on all hosts that belong to the distributed system. Define the new directory as a network drive.



```
[mappings]
map_t=\\myfileservice\myoldtrexshare
map_u=\\myfileservice\mynewtrexshare
```

For more information, see [Defining the Network Drive \[Page 34\]](#).

3. Stop TREX on all hosts that belong to the distributed system. Restart TREX.
4. Start the TREX admin tool on any host in the distributed system.
5. Go to the *Landscape Configuration* window.
6. Specify the new network drive, or a subdirectory thereof, for all hosts in the *Basepath* column of the *Hosts* table.
7. Activate this change by choosing *Deploy*.

## Procedure with Decentralized Data Storage

1. Create a new TREX data directory on the host in question.
2. UNIX only: Make sure that the directory belongs to the user `SAPService<SAPSID>`.
3. Start the TREX admin tool on any host in the distributed system.
4. Go to the *Landscape Configuration* window.
5. Specify the new directory for the affected host in the *Basepath* column of the *Hosts* table.
6. Activate this change by choosing *Deploy*.



## Distributed Preprocessing of Documents

### Purpose

Indexing is a complex process consisting of several phases. One phase is the preprocessing of documents by the preprocessor. Preprocessing includes the following steps:

- Loading documents if the application transmitted them as URIs.
- Filtering
- Carrying out a linguistic analysis

Preprocessing can take a similar amount of time and use similar system resources to the actual indexing process. The filtering of a large number of large documents that are not in text or HTML form can be particularly time- and resource-consuming (for example, large PDFs).

In order to increase throughput in preprocessing, you can distribute the preprocessing among multiple hosts. For example, you can use one host (or more than one) exclusively for preprocessing documents. You do this if there are a large number of documents to be preprocessed for the initial indexing run.

The following sections contain information on the distributed preprocessing of documents.

- The section [Fundamentals \[Page 87\]](#) explains the preprocessing flow for indexing. It also tells you about distribution options and how to control load distribution and performance.
- The section [Configuration \[Page 94\]](#) explains how to configure distributed preprocessing.



The preprocessor is involved in processing search and text-mining requests as well as in indexing. In all of these processes, the preprocessor has the task of preparing the actual preprocessing.

The sections below only relate to the preprocessing of documents for indexing. The role of the preprocessor in processing search and text-mining requests is not described.



## Fundamentals

The following sections provide fundamental information on the topics below.

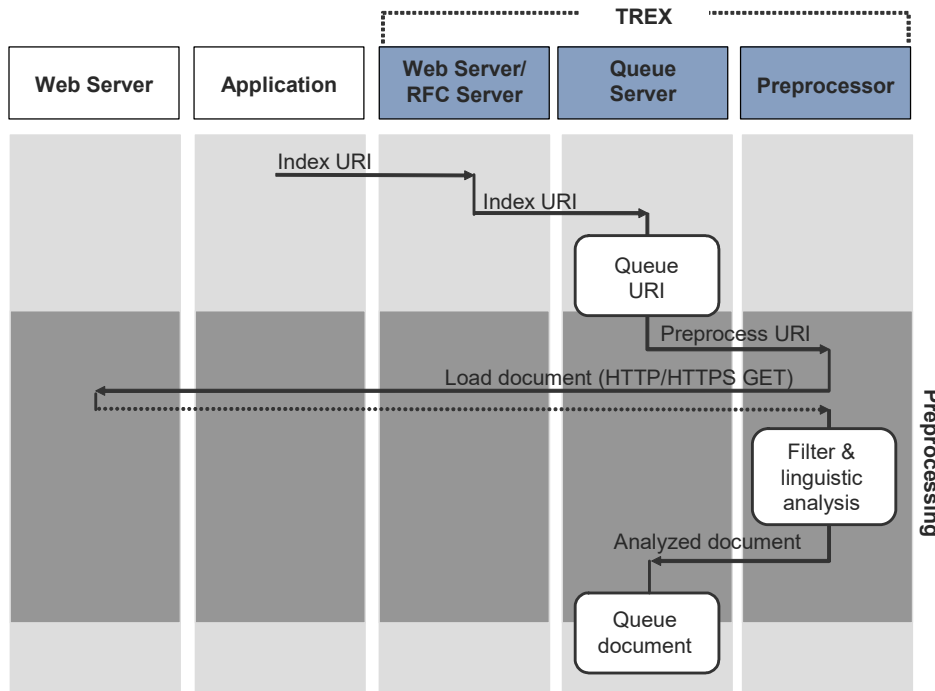
- [Preprocessing Flow \[Page 87\]](#)
- [Distributing Preprocessing \[Page 89\]](#)
- [Load Distribution and Performance \[Page 89\]](#)



## Preprocessing Flow

The graphic below depicts the most important steps that take place immediately before, during, and after preprocessing.

The graphic depicts the process flow of the application transmitting the *URI* of a document to TREX. If the application transmits the document directly, the step 'Load document (HTTP/HTTPS Get)' does not take place.



The application sends indexing requests to the TREX Web server or TREX RFC server. This server then forwards the requests to the queue server. The queue server assigns the requests to the correct queues and distributes the requests among one or more preprocessors. The actual preprocessing of documents then takes place on the preprocessor(s).

When the preprocessing has been completed, the preprocessor passes the analyzed document to the queue server. The queue server collects the documents and, depending on its configuration, triggers further processing on the index server.

## How Does the Distribution of Documents Take Place?

The distribution of documents among the preprocessors is controlled by the name server. The distribution takes place according to a round robin procedure that takes the number of times that a preprocessor has been accessed into account. Preprocessors that have been accessed less often are preferred when distributing documents.

The process flow is as follows:

1. When a queue server receives a document it assigns it to a preprocessor client.
2. The preprocessor client asks the name server for the address of a preprocessor.
3. The name server returns the preprocessor that has been accessed least often.
4. The preprocessor client forwards the document to the preprocessor and waits for a response. Preprocessor clients are busy while waiting for a response. They receive no further documents from the queue server during this time.
5. When the preprocessing of the documents is over, the preprocessor client receives a response from the preprocessor, and returns its own response to the queue server.
6. Only then is the preprocessor client free to receive further documents from the queue server.



## Distributing Preprocessing

The preprocessing of documents is carried out by preprocessors running in `any` or `index` mode. If you set up the system according to [Landscape Configuration \[Page 37\]](#), these are

- The preprocessors that run on the master hosts
- If you are using backup hosts, the preprocessors that run on the backup hosts



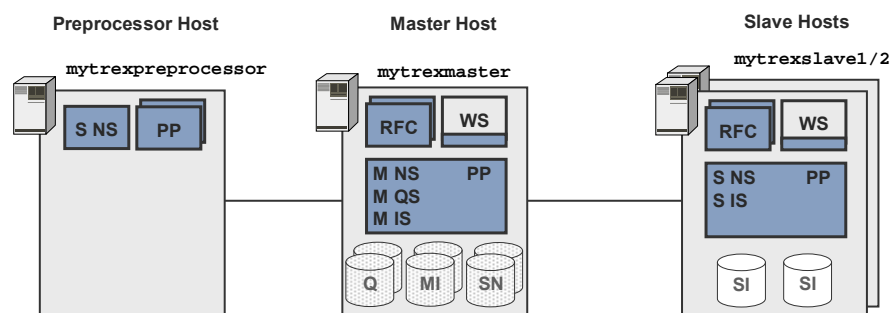
For more information on the meaning of the modes, see [Preprocessor Modes \[Page 9\]](#).

If the preprocessing capacity of the master and backup hosts is insufficient, you can use one host or multiple hosts exclusively for preprocessing. Preprocessing then takes place on additional preprocessors, allowing more documents to be preprocessed in parallel. This increases throughput for preprocessing.

On a host used exclusively for preprocessing, one or more preprocessors run in `index` mode, and a name server also runs. Such a host is referred to as a preprocessor host.



The graphic below depicts a system with one master host, two slave hosts, and one preprocessor host. The preprocessor host supports the master host in preprocessing.



## Load Distribution and Performance

You have to configure the preprocessors and queue server if you want to use distributed preprocessing. The following parameters are important for load distribution and performance:

- [Number of Preprocessors and Preprocessor Threads \[Page 90\]](#)
- [Preprocessor Threads and Queue Server Pool Size \[Page 92\]](#)

You can only improve performance by taking all parameters into account together. Changing one individual parameter cannot improve performance.





## Number of Preprocessors and Preprocessor Threads

The following parameters in the preprocessor and queue server are important for performance:

- The number of preprocessors running on a host  
(Number of preprocessors per host)
- The number of threads in a preprocessor process  
(Number of threads per preprocessor)
- Number of preprocessor clients in the queue server  
(Pool size per queue server)



You can use the pool size for the queue servers to directly influence the number of preprocessor threads. The number of preprocessor threads and the pool size are connected as follows: `<queue server pool size> = <number of preprocessor threads>` For more information, see [Preprocessor Threads and Queue Server Pool Size \[Page 92\]](#).

### Configuration Rules for Preprocessor and Queue Server

You must take into account the following relationships and configuration rules for a high-performance configuration of distributed preprocessing:

- `<maximum number of preprocessors per host> = <number of CPUs>`  
That is, a maximum of one preprocessor per CPU.
- `<maximum number of threads per preprocessor> = 3`  
That is, a maximum of three threads per preprocessor and per CPU.
- `<total pool size of all queue servers> = <total number of CPUs for all preprocessor hosts> * 3`

These relationships are explained in more detail below.

### How Many Preprocessors Can Run On a Host?

The number of preprocessors that can run on a host is limited by the available main memory and the number of CPUs.

Each preprocessor process has its own main memory area. If there are multiple preprocessors running, they need a correspondingly large amount of main memory. The main memory requirement of a preprocessor depends on the following factors:

- How big are the documents?
- What format do the documents have (PDF, HTML, and so on)?
- For how many languages is language recognition activated?

The main memory requirement for one language is between 30 and 40 MB per preprocessor. If there are more languages, the main memory requirement is normally around 100 MB per preprocessor.

In some cases, the main memory requirement may be between 500 MB and 1 GB. The worst case scenario can occur if language recognition is activated for all languages and a large number of preprocessor threads are processing large documents at the same time.

If the host has enough main memory, the following upper limit is valid:

`<Maximum number of preprocessors on a host> = <number of CPUs>`

## What Is the Maximum Possible Number of Preprocessor Threads?

A preprocessor process can consist of one or more threads. If there are multiple threads, the preprocessor can distribute the requests among the threads and process the requests in parallel. The preprocessor automatically starts the number of threads that is required for processing.

For each preprocessor process, a maximum of three preprocessor threads per CPU should be started:

`<number of preprocessor threads per preprocessor process> = 3`

Since only one preprocessor per CPU and only three threads per preprocessor should be started, this results in the following relationship:

`<maximum number of all preprocessor threads running on a host> =  
<number of CPUs> * 3`



You use the queue server pool size to indirectly configure the number of preprocessor threads (see [Preprocessor Threads and Queue Server Pool Size \[Page 92\]](#)).

If the preprocessor uses the maximum number of threads it is also using the maximum amount of system resources. You will have almost complete CPU load.



If you want the preprocessor to have fewer system resources, you can choose to have a smaller number of threads. However, you ought not to choose to have a greater number of threads, since this can cause performance to drop.

The more threads invoked in parallel, the longer the operating system takes to administrate the threads (to trigger, stop, and monitor them). If the number of threads invoked in parallel is too great, the operating system is overwhelmed by thread administration.

## More Preprocessors or More Threads?

If you want to optimize preprocessing performance, you need to decide whether to increase the number of preprocessors or the number of preprocessor threads. Your decision depends on the following factors:

- Required load distribution among the hosts
- System resources of the hosts (number of CPUs and available main memory)

If only **one** host is preprocessing documents, it makes no difference whether one preprocessor is running with multiple threads or several with one thread each.

If several hosts are preprocessing documents, the parameters have the following effect:

- **Load balancing**

The number of preprocessors running on each host controls the load distribution among the hosts.

The more preprocessors running on a host, the more load that host receives.



Preprocessing takes place on the master host and on a preprocessor host. Because the master host also carries out indexing you want it to receive a smaller preprocessing load. There is therefore only one preprocessor on the master host, but two preprocessors on the preprocessor host.

The load is distributed among the two hosts in the ratio 1:2.

- **Performance**

The number of preprocessor threads controls the performance on one host.

The more threads there are, the more documents a preprocessor can process in parallel.

You cannot use the pool size on the queue server to increase the number of preprocessor threads (see [Preprocessor Threads and Queue Server Size \[Page 92\]](#)) and the number of preprocessors without restriction. The maximum number depends on the available system resources.

## Availability

Availability can also play a part when deciding on the number of preprocessors and preprocessor threads.

Using multiple preprocessors increases the availability of the system. This is because different processes (preprocessors) have less impact on one another than do the different threads of a process. If a thread hangs, this can affect other threads of the same process but not of another process.

However, using multiple preprocessors also requires more main memory (see the *How Many Preprocessors Can Run On a Host?* section above).



## Preprocessor Threads and Queue Server Pool Size

The pool size is important for achieving optimum integration between the queue servers and preprocessors. The pool size determines how many documents a queue server can distribute to the preprocessors at once.

From a technical point of view, the pool size determines how many preprocessor clients a queue server instantiates at startup. The preprocessor client is an internal component of the queue server. The queue server uses the preprocessor clients to communicate with the preprocessors and uses its services.

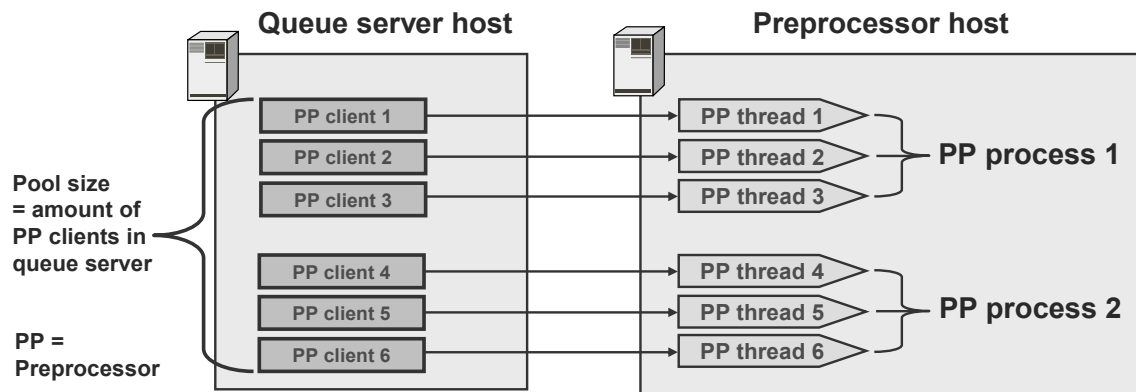
Depending on the number of preprocessor clients started in the queue server (= pool size), the corresponding number of preprocessor threads are started by a central worker thread management. You can use the pool size in the queue server to control the number of preprocessor threads on the hosts that preprocessors are running on.

The following relationship applies:

`<queue server pool size> = <number of preprocessor threads>`

## Example

For example, if you set the pool size in the queue server to the value 6, the corresponding number of preprocessor threads are started on the host that the preprocessor(s) are running on. If two preprocessor processes are running there, the threads are distributed between two preprocessor processes, which correspond to three threads per process.



**Example of the relationship between the pool size and the number of preprocessor threads**

## What Value Should the Queue Server Pool Size Have?

You must determine the optimum pool size and thus the number of preprocessor threads individually for your system. For each preprocessor process, a maximum of three preprocessor threads should be started using the entry for the pool size.

Thus, the following relationship applies:

`<number of preprocessor threads per preprocessor process> = 3`

Since a maximum of only one preprocessor should be started per CPU (see [Number of Preprocessors and Preprocessor Threads \[Page 90\]](#)), this results in the following relationship for a distributed system landscape with multiple queue servers and preprocessor hosts:

`<total pool size of all queue servers> =  
<total number of CPUs for all preprocessor hosts> * 3`

If the pool size is too low, the preprocessor can have unnecessary idle times and not have a full load, although resources are still available. If the pool size is too large, the host on which the queue server is running uses too many system resources to manage the pool.

You should check the CPU load for the preprocessors for a while. If system resources are still available, you can increase the pool size to improve performance. However, if you increase the pool size beyond the recommendations, you gain no performance benefits and might actually cause performance to drop.

The pool size of queue servers is configured in the file `TREXQueueServer.ini`.



## Configuration

### Purpose

The sections below explain how to set up distributed preprocessing with a preprocessor. It also contains information on how to increase the number of preprocessors and preprocessor threads if necessary.



## Configuration Recommendations

To achieve high-performance preprocessing that does not hamper the other TREX servers, use the following configuration.

### Preprocessor hosts

- In accordance with the configuration rules that are specified in [Number of Preprocessors and Preprocessor Threads \[Page 90\]](#), start the required number of preprocessors on the preprocessor host.
- Monitor the load on the host during preprocessing. If system resources are still available, you can use the pool size in the queue server to increase the preprocessor threads up to the maximum number recommended.

### Master Host

We recommend that you keep the default configuration for the preprocessor on a master host.

If you give the preprocessor additional system resources, the performance of the queue server and index server suffers. Preprocessing will be faster, but subsequent processing steps will be slower.

### Backup host

If the master index server and master queue server are active, there is little load on the backup hosts. If you want to use more load for preprocessing on a backup host, you can start more preprocessors on it, provided the hardware allows this (when doing this, note the configuration rules that are specified in [Number of Preprocessors and Preprocessor Threads \[Page 90\]](#).) This allows you to make better use of the system resources on the backup host. However, the performance of the indexing is not so high if either the backup index server or backup queue server is actually active.

## Example

### Example 1

The following hosts preprocess documents:

- Master host:  
2 CPUs – one preprocessor – one queue server
- Preprocessor hosts:  
2 CPUs – two preprocessors

Both hosts have two CPUs. Because the preprocessor host **only** preprocesses documents, it should take more of the load than the master host. The preprocessor host therefore has two preprocessors. The only queue server in the system and a preprocessor are running on the master host. Therefore, a total of three preprocessors are running. Since each preprocessor process should process a maximum of three threads per CPU in parallel, you can calculate the maximum pool size as follows:

`pool size = number of all preprocessors * 3 = 3 * 3 = 9`

Thus the pool size must be set to the value 9 in the configuration file `TREXQueueServer.ini` on the master host. As a result, the queue server makes available a total of nine queue server clients for the preprocessors and in turn a total of nine threads are started in the preprocessors.

### Example 2

The following hosts preprocess documents:

- Master host 1  
2 CPUs – one preprocessor – one queue server
- Master host 2  
2 CPUs – one preprocessor – one queue server
- Backup host  
2 CPUs – one preprocessor – one queue server
- Preprocessor host  
2 CPUs – two preprocessors

The preprocessor host therefore has 2 preprocessors, as in example 1, but no queue server. Since the system consists of two master hosts and one backup host, there are a total of three queue servers. We can assume that two of these three queue servers are always active: Either both master queue servers, or one master queue server and one backup queue server.

The pool size for two active queue servers is determined as follows:

`pool size = number of all preprocessors * 3 = 5 * 3 = 15`

This pool size divided by the number of active queue servers gives a pool size of 7 or 8 per queue server. This is the pool size that you enter in the configuration file `TREXQueueServer.ini` of all queue servers.



## Setting Up Distributed Preprocessing

### Use

The procedure below explains how to implement distributed preprocessing. The description assumes that:

- You have set up a distributed system with at least one master host.
- You want to connect a host that exclusively preprocesses documents (preprocessor host). You want the preprocessors on this host to have as many system resources as possible.

## Adding a Preprocessor Host to the Distributed System

1. Install TREX on the preprocessor host. During the installation specify the number of preprocessors to run on the host.
2. If TREX is not running, start it.
3. Start the TREX admin tool on a host that is already configured in the distributed system.
4. Go to the *Landscape Configuration* window.
5. Use *Add Host* to add the new preprocessor host.

## Configuring Preprocessor Hosts

1. Choose the preprocessor mode *index* for the preprocessor host.
2. Configure the TREX daemon on the preprocessor host so that only the name server and preprocessors run there:
  - a. Select the host in question and choose *Edit Services*.
  - b. Change the `programs` parameter as follows:

```
[daemon]
programs = nameserver, preprocessor1, ..., preprocessor<n>
```
3. Go to the *Landscape Services* window.
4. Select one of the servers to run on the preprocessor host. Choose *Start New/Stop Removed Services @<hostname>(\*)* from the context menu.

## Configuring Master and Backup Hosts

1. Go to the *Landscape Ini* window.
2. Establish the maximum possible number of preprocessor threads for all hosts that preprocess documents. Take into account all hosts on which a preprocessor is running in either *any* or *index* mode.

For more information about the calculation, see [Preprocessor Threads and Queue Server Pool Size \[Page 92\]](#).
3. Calculate the pool size for each queue server.

For more information, see [Preprocessor Threads and Queue Server Pool Size \[Page 92\]](#).
4. Edit the configuration file `TREXQueueServer.ini` for all queue servers. Enter the calculated value in the parameter `poolsize`.
5. Go to the *Landscape Services* window.
6. Select a queue server whose configuration you have changed. Choose *Restart queueserver @<host\_name>:<port>* from the context menu.

Carry out this step for all other queue servers.

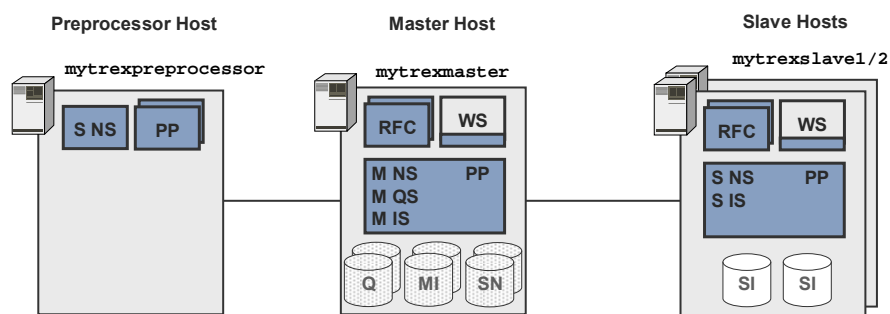
The queue servers are automatically restarted by the TREX daemon.

## Result

You can check whether the preprocessors are receiving as many system resources as possible by looking at the CPU load for the hosts in question in the TREX admin tool. When documents are being preprocessed, the CPU usage should be at the upper limit.

## Example Configuration

This section shows the configuration for a system in which preprocessing takes place on one master host and one preprocessor host.



The configuration is only specified for `mytrexmaster` and `mytrexpreprocessor`, and only where distributed preprocessing is involved.

Assumptions:

- Both hosts have 2 CPUs each.
- Two preprocessors should run on `mytrexpreprocessor`.
- The only queue server in the system is running on `mytrexmaster`.
- Only searches take place on `mytrexslave1/2`, there is no preprocessing here.

TREX admin tool, *Landscape Configuration*

### Hosts table (extract 1)

Host	Name Server Mode	Master Index/Queue Server	Slave Index Server for	Preprocessor Mode
<code>mytrexmaster</code>	<i>1st master</i>	✓		<i>index</i>
<code>mytrexpreprocessor</code>	<i>slave</i>			<i>index</i>
<code>mytrexslave1/2</code>	<i>slave</i>			<i>search</i>
...				

### Hosts table (extract 2)

Host	Base Path	Services
<code>mytrexmaster</code>	...	...
<code>mytrexpreprocessor</code>	<code>/usr/sap/&lt;SAPSID&gt;/TRX&lt;br&gt;&lt;instance_number&gt;</code>	<code>nameserver, preprocessor1, preprocessor2</code>
...		

### TREXDaemon.ini for 'mytrexpreprocessor' (extract)

```
[daemon]
programs = nameserver, preprocessor1, preprocessor2
```



TREX admin tool, *Landscape Ini*

### TREXQueueServer.ini for 'mytrexmaster'

```
[preprocessor]
```

```
poolsize=9
```

The pool size is calculated as follows:

$$2 * \text{<preprocessor-threads>} \text{ auf } \text{<mytrexpreprocessor>} + \text{<preprocessor-threads>} \text{ on } \text{<mytrexmaster>} = 2 * 3 + 3 = 9$$


## Increasing the Number of Preprocessors

### Use

If necessary you can increase the number of preprocessors running on a host. See [Number of Preprocessors and Preprocessor Threads \[Page 90\]](#) for information on when this is recommended.

### Procedure

1. Start the TREX admin tool on any host in the distributed system.
2. Go to the *Landscape Configuration* window.
3. Select the host in question and choose *Edit Services*.
4. Add an entry for the new preprocessor to the parameter `programs`.  

```
[daemon]
programs = ..., preprocessor<new_number>
```
5. Make sure that there is a section with the same name (`preprocessor<new_number>`) containing the start parameter for the preprocessor. If there is no such section, copy an existing section and rename it as follows:  

```
[preprocessor<new_number>]
Windows: executable=TREXPreprocessor.exe
UNIX: executable=TREXPreprocessor.x
. . .
```
6. Go to the *Landscape Services* window.
7. Select any TREX server running on the host in question. Choose *Start New/Stop Removed Services @<hostname>(\*)* from the context menu.
8. Modify the pool size of all master and backup queue servers.

For information on calculating the pool size, see [Pool Size of Queue Servers \[Page 92\]](#).  
 For information on the procedure, see the section *Master and Backup Hosts* in [Setting Up Distributed Preprocessing \[Page 95\]](#).



## Appendix



### Information on Stopping/Starting Distributed Systems

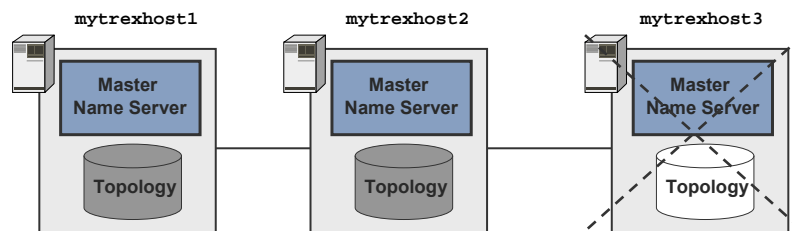
There are no special rules to take into account when stopping a distributed system. You can stop TREX in any order on the individual hosts.

When you start a distributed system, the type of data storage dictates whether there is a defined sequence.

- If you are using centralized data storage, there is no special sequence.
- If you are using decentralized data storage, you firstly have to start a master name server that was running just before the system was stopped. This ensures that the system is based on an up-to-date topology file.



The hosts `mytrexhost1`, `mytrexhost2`, and `mytrexhost3` are configured as master name servers. `mytrexhost3` has not been operating for a while, which means that its topology file is not up-to-date. Changes that have been made since (such as new indexes) are not known to this host.



You also stop TREX on the remaining hosts for maintenance reasons. If you now want to restart TREX, you now have to start it on `mytrexhost1` or `mytrexhost2` first. These master name servers have up-to-date topology files.

If you were to start TREX on `mytrexhost3` first, the system would be based on an out-of-date topology file.



The master name servers compare their topology files at startup. If the files are different, the master name server saves the files as `topology.<date>.old` and `topology.<date>.new`. This allows the correct topology to be restored even if the required start sequence is not observed.

If this happens in your system, contact SAP Support.




## Starting the TREX Admin Tool

### Prerequisites

On UNIX: Since the TREX admin tool has a graphical interface, you need an X server. You cannot use a terminal program that only supports text mode, such as `telnet`.

### Procedure

1. Log on with the user `<sapsid>adm`.
2. Carry out one of the following steps:

Operating System	Procedure
UNIX	Enter the following: <pre>cd &lt;TREX_DIR&gt; ./TREXAdmin.sh</pre>
Windows	Choose <i>Start → Programs or All Programs → SAP TREX → Instance &lt;instance_number&gt; → Tools → TREX Administration</i>  You can also start the TREX admin tool by double-clicking <code>&lt;TREX_DIR&gt;\TREXAdmin.bat</code> in Windows Explorer.



## Configuring Queue Parameters

### Use

The queue parameters control the interaction between the queue server and the index server. In particular, they specify when the queue server triggers indexing and optimization of documents. It is important for performance reasons that you have optimum settings for the queue parameters.

When TREX creates a queue, it uses the default settings for the queue parameters. Depending on the document sets that you have to index initially and on the type of documents you index, you may have to change the default settings.



The default settings that TREX uses for new queues are defined in the configuration file `TREXQueueServer.ini`. You can change the default settings. However, you should only make changes to configuration files after consulting SAP support or with a consultant.

### Prerequisites

You have already created indexes.

## Procedure

You can change the queue parameters for existing queues as follows:

Tool	Path
TREX admin tool	<i>Queue Admin → Queue Parameters</i>
TREX monitor in the portal	<i>System Administration → Monitoring → Knowledge Management → TREX Monitor → Edit Queue Parameters</i>
TREX Admin Tool in the SAP System	<i>Transaction TREXADMIN → Queue Admin → Set Queue Parameters</i>

For more information about the meaning of the queue parameters, see the SAP Library at [help.sap.com](http://help.sap.com).



## Changing Java Client Parameters

### Use

You change the Java client parameters using the *SAP J2EE Engine Visual Administrator Tool*.

### Procedure

1. Log on to the host on which the SAP J2EE Engine is running. Use the user `<j2eeadm>`.
2. Start the *SAP J2EE Engine Visual Administrator Tool* and log on to the SAP J2EE Engine.

For information on using this tool, see the SAP Library at [help.sap.com](http://help.sap.com).

3. Choose *Cluster → Services → TREX Service*.
4. Make the required changes.
5. Save your changes and confirm the restart of the service.
6. Repeat the last three steps for all other server processes of the cluster.