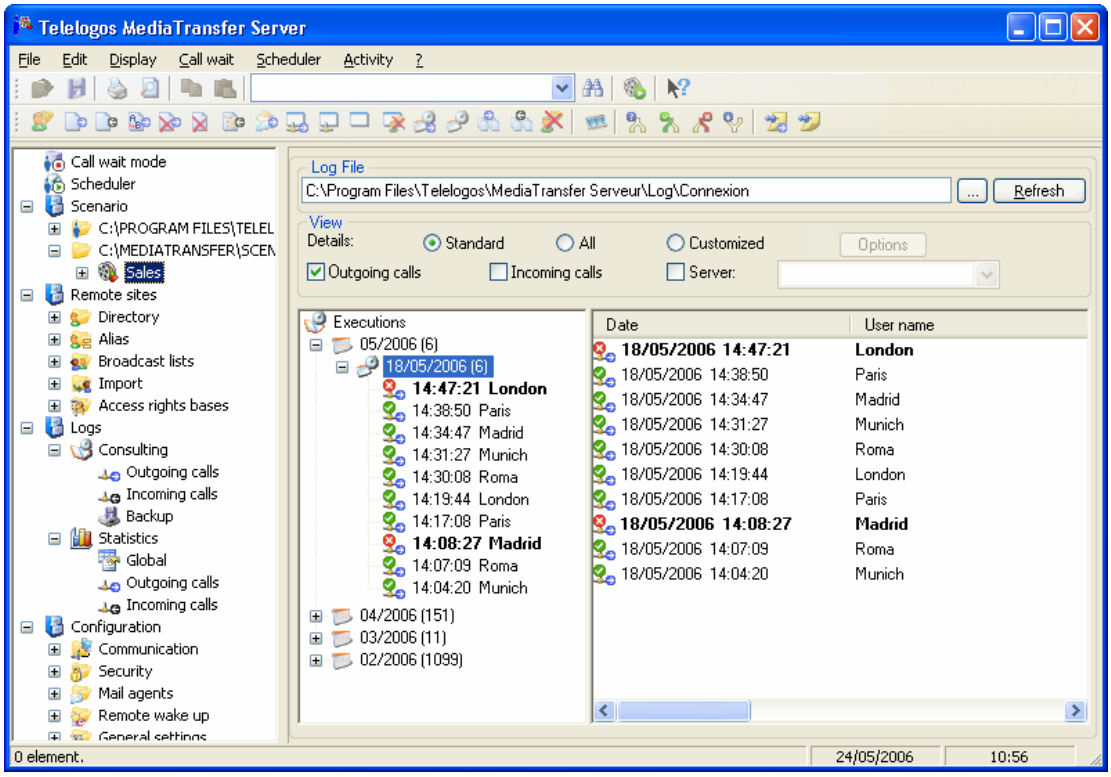
MEDIATRANSFER

Version 6.0

For Windows Vista, 2003, 2000, NT, XP,
Windows Mobile 5.0 / 2003, CE,
Linux Red Hat & SuSE

TECHNICAL GUIDE



TELELOGOS
Linking people to companies

3, Avenue du Bois l'Abbé – Angers Technopole – 49070 Beaucouzé – France
Tel: +33 (0)2 41 22 70 00 – Fax +33 (0)2 41 22 70 22

www.telelogos.com

TABLE OF CONTENTS

Introduction

MediaTransfer provides powerful automation solutions that let businesses automate, secure and optimize communication between their information system and remote machines (terminals) or computers used by mobile and sedentary workers.

MediaTransfer automates the calls, connections, transfers and synchronization of all file types and tree structures, the synchronization of messaging systems (Microsoft Outlook or Lotus Notes) and the launch of programs. Developers can use MediaTransfer to integrate automation features for data flow between servers and remote or mobile computers into the programs they create or maintain. MediaTransfer controls data flow via intermittent communications. MediaTransfer integrates with all application types and enables the simultaneous transfer of application and messaging data during a single communication session. The software optimizes communication costs and times using online compression, automatic failure recovery and differential synchronization. MediaTransfer empowers administrators by providing them with a simple means to control communications using scenarios. It also lets them monitor and analyze communications via logs and statistical information.

MediaTransfer Functions

- Automates connections, data flows and remote processing
- Operates transparently - mobile and sedentary workers can focus on their main tasks
- Integrates into automatic machines (terminals)
- Integrates seamlessly and rapidly with any application
- Integrates with Lotus Notes, Microsoft Outlook and Microsoft Outlook Agents
- Symmetrical architecture: the server can call the client and the client can call the server
- Inter-server communications for division-based architectures
- Up to 512 simultaneous channels, distributed across several communication servers, each managing up to 256 channels
- Integrates with TCP/IP architectures (Internet, intranet) and asynchronous architectures
- Optimizes telecommunications costs and time using online compression, checkpoint restart and differential synchronization
- Optimizes the load of IP networks through dynamic bandwidth management
- Secures data transfers through the use of the SSL protocol (128-bit AES encryption and authentication by certificates)

MediaTransfer Architecture

MediaTransfer Server

Each MediaTransfer Server includes:

- An **administration console**
- An **automation manager to execute communication scenarios**
- A **scheduler** to launch scenarios
- A driver that manages up to 256 **simultaneous connections**
- **APIs** (Application Programming Interface)
- A **database** (Microsoft Access, Microsoft SQL Server**,** Oracle or MySQL) that stores statistics and logs

MediaTransfer Server can coexist on the same local network as:

- An **administration console**
- A **communications monitor**

Notes:

The server, console and monitor can reside on a single machine or on several machines in the same local network.
Flat files can replace the database server.

Minimum configuration for the server:

- Windows 2003, 2000, XP
- Disk: 100 MB
- Memory: 32 MB + 2 MB per line

Minimum configuration for the administration console and communications monitor:

- Windows 2003, 2000, XP

MediaTransfer Clients

MediaTransfer **Client** includes the following on each remote computer:

- An automation manager to execute communication scenarios
- A scheduler to launch scenarios
- A driver that manages 1 communication line
- APIs (Application Programming Interface)
- A GUI (Graphical User Interface)

Notes:

MediaTransfer Clients can only communicate with MediaTransfer Server.

Configurations supported by MediaTransfer **Client** version 6.0:

- Windows Vista, 2003, 2000, NT, XP
  - Disk: 100 MB; Memory: 16 MB
- Windows Mobile 5.0 / 2003, CE
  - Program memory used: 1 MB

Configurations supported by MediaTransfer **client** version 5.0:

- Windows 9x
  - Disk: 40 MB; Memory 7 MB
- Linux Red Hat / SuSE

A “light” client (no user interface) can also be installed to facilitate remote distribution.

A Network

MediaTransfer operates on an **IP network** to establish TCP/IP connections between MediaTransfer Server and remote MediaTransfer Clients.

In most instances MediaTransfer Server is installed on a company’s local network and has IP access via a router (or access server) connected to a public network (ISDN, etc.) or an

Intranet. Depending on your security needs, MediaTransfer Server can be placed upstream or downstream of your organization's firewall.

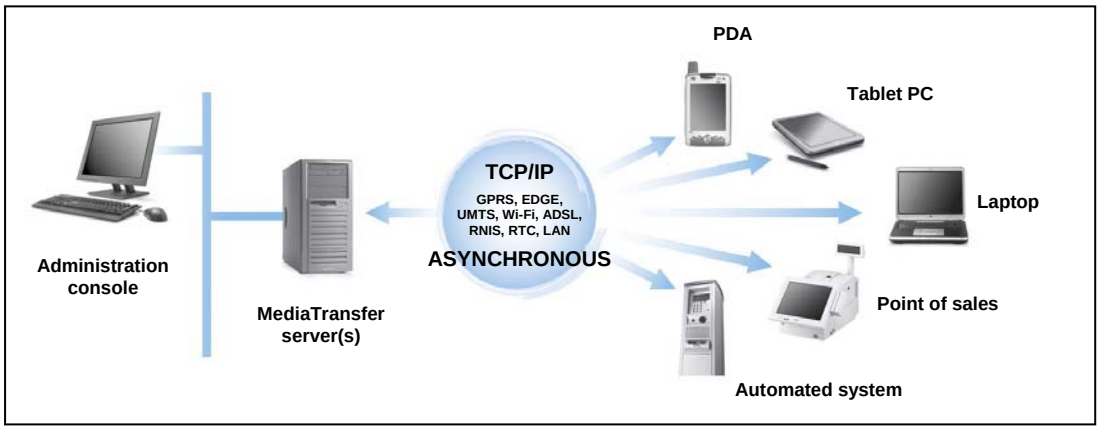
MediaTransfer Clients can be installed on the same local network as the Server (for testing, to validate the architecture, etc.). Remote stations are usually connected to a public network: PSTN, ISDN, GPRS, GSM, ADSL, Wi-Fi…

According to how MediaTransfer Server is integrated into your network, connections between the server and each client can either be point-to-point or pass through an intermediate server (Internet, shared Intranet maintained by a provider, etc.).

MediaTransfer can also operate on an **asynchronous point-to-point network** to establish an asynchronous link between MediaTransfer Server and remote MediaTransfer Clients (PSTN, ISDN networks).

Whether or not the call can be initiated or established by a MediaTransfer Server and/or MediaTransfer Client depends on the characteristics of the network.

Note: Windows CE and Pocket PC require **TCP/IP communications**.



Operating Principles

MediaTransfer is installed on one or several servers and on each remote station. The structures of the server and client software are identical:

- A driver controls communication lines (up to 256 lines per server, 1 line for each client)
- The program automatically executes commands saved in scenarios

The server and clients can be in one of three states at any given time:

- Waiting
- Executing a scenario
- Receiving a call

A scenario is a structured file that contains telecommunication commands.
Executing a scenario on the server lets the server call one or more remote clients using one or more communication lines. The client switches to the **Receiving** call state when the call is received. **Executing** a scenario on the client lets the client call a server. The server line receiving the call switches to the **Receiving** call state. When **Execution** of the scenario has ended, the server or client returns to the **Waiting** state and waits for another scenario to be executed or to receive a call.

The server and the client are differentiated by the number of communication lines they have. At any one moment, each line on the server can be caller, callee, or waiting, regardless of the states of the other lines.

The caller and callee log all scenario execution events. Log information can be viewed, printed and utilized for different purposes.

Scenario execution can be started in three different ways:

- By a server or client application via MediaTransfer's API
- By the MediaTransfer scheduler on the server or client
- By the server administration console or the client GUI

Launch settings let you customize execution of each scenario (number of retries if failure occurs, amount of time to wait before retry, etc.).

MediaTransfer's administration console user interface makes building and managing scenarios easy. Tests, conditions, jumps and variables give your scenarios all the power they need to carry out the actions you require.

Remote contacts are listed in directories for the server and each remote client. Their access rights are managed in rights databases. The server's contacts directory can be synchronized with an external data source (relational database, LDAP directory, ODBC accessible data source).

Scenario

A scenario contains a set of commands that lets a computer (either a server or a client) call one or several remote computers and carry out transfers and/or perform processing. During a call between two computers (a server and a remote client), the scenario piloting the call is executed by the calling computer.

A scenario executed on a client may contain only a call command with a setting that informs the server it is calling in slave mode. The server then retrieves the scenario name for this client from the directory and executes it.

Several scenarios can be executed simultaneously, and they may be assigned an order of priority to optimize data flow.

List of commands:

- Call one or several contacts
- Send one or several files
  - Backup local files before sending
  - Backup remote files after sending
  - Conditional delete of local file
- Retrieve one or several files
  - Backup remote files before retrieving
  - Backup local files after retrieving
  - Conditional delete of remote file
- Synchronize files or tree structures
- Delete one or several local files
- Delete one or several remote files
- Rename a remote file
- Request the list of files in a remote folder
- Sending an SMTP message
- Display a message on the screen of the local or remote computer
- Launch remote execution
- Launch local execution before call, during call, after call

- Execute messaging agents
- Synchronize the caller's system date/time with the callee's
- Synchronize the callee's system date/time with the caller's
- Execute a sub-scenario
- Evaluate a condition
- Test a condition
- Launch a local execution at the end of a call, with or without the condition that call commands must have been properly executed

Scenario Commands by Feature

Call:

Objective: Call one or several contacts.

Main Settings:

- Contact's access *number*
- Contact's *name* from the callees directory
- Name of *contacts list file* containing a list of contacts from the callees directory

Send one or several files:

Objective: Send one or several files from the local computer to the remote computer.

Main Settings:

- Name of *local file* (with or without absolute or relative path)
- Name of *remote file* (with or without absolute or relative path)

or

- Name of *list file* containing a list of local files to send

Other settings:

- Continue scenario execution if the local file does not exist
- Continue scenario execution if the local file is locked
- Send files in "append mode" in the remote file
- Do not send the file if it is already present on the remote computer
- Retry send during a fixed period of time as long as the local file cannot be accessed
- Set the order of files when sending multiple selections
- Delete the local file when it is sent successfully
- Lock the local file during its transfer

When writing in append mode, it is possible to have the same number of files on the remote computer as files sent using numbering.

Backup local files before sending

Objective: Backup files on the local computer before transferring them to the remote computer. This command immediately precedes the **Send** command.

Backup remote files after sending

Objective: Backup files on the remote computer after sending them. This command immediately follows the **Send** command.

Conditional delete of local files

Objective: Delete the local file when it is sent successfully. This command is associated with the **Send** command.

Retrieve one or several files

Objective: Retrieve one or several files from the remote computer and have it (them) sent to the local computer.

Main Settings:

- Name of *local file* (with or without absolute or relative path)
- Name of *remote file* (with or without absolute or relative path)

or

- Name of *list file* containing a list of remote files to retrieve

Other settings:

- Continue scenario execution if remote file does not exist
- Continue scenario execution if the remote file is locked
- Retrieve files in “append mode” in the local file
- Do not retrieve file if it is already present on the local computer
- Retry retrieve during a fixed period of time as long as the remote file cannot be accessed
- Set the order of files when retrieving remote files using multiple selections
- Delete the remote file when it is transferred successfully
- Lock the remote file during its transfer

Backup remote files before retrieving

Objective: Backup files on the remote computer before transferring them. This command immediately precedes the **Retrieve** command.

Backup local files after retrieving

Objective: Backup files on the local computer after retrieving them from the remote computer. This command immediately follows the **Retrieve** command.

Conditional delete of a remote file

Objective: Delete the remote file if its retrieval successful. This command is associated with the **Retrieve** command.

Synchronize files or tree structures

Objective: Synchronize files or tree structures between two remote computers. Make a set of files in a folder on a target system identical to a set of files in a folder on a reference system.

Only new or different files are transferred from the reference system to the target system. The reference system can either be the remote or the local computer.

Delete one or several remote files

Objective: Delete one or several remote files.

Settings:

- Name of *remote file* (with or without absolute or relative path)

or

- Name of *list file* containing a list of remote files to delete

Delete one or several local files

Objective: Delete one or several local files.

Settings:

- Name of *local file* (with or without absolute or relative path)

or

- Name of *list file* containing a list of local files to delete

Rename a remote file

Objective: Rename a remote file.

Settings:

- *New name* of remote file (without absolute or relative path)

or

- Name of *list file* containing a list of remote files to rename and their new names

Request folder

Objective: Request the list of files in the remote folder. The list is transferred in a file. There are two types of lists:

- Simple list: File names and sizes
- Extensive list: File names, sizes, date and time files where saved

Settings:

- Name of *local list file* (with or without absolute or relative path)
- Type of list

Sending an SMTP message

Objective: Send an e-mail to a contact

Settings:

- SMTP server name
- SMTP port number
- Sender's e-mail address
- Recipient's e-mail address
- Subject of the message
- Text of the message
- Name of any attached file

Display a message on the screen of the local computer

Objective: Open a message box on the screen of the computer running the scenario.

Settings:

- Message box *title*
- Text of the message

or

- Name of *local file* (with or without absolute or relative path) that contains the message to display

Display a message on the screen of the remote computer

Objective: Open a message box on the screen of the called computer

Settings:

- Message box *title*
- Text of the message

or

- Name of *local file* (with or without absolute or relative path) that contains the message to display

Launch a remote execution

Objective: Launch the execution of a program, system command or executable file on the remote computer. There are two types of execution:

- Asynchronous: Execution is launched on the remote computer and the scenario goes to the next command without waiting for the execution to complete
- Synchronous: Execution is launched on the remote computer and the scenario waits (during a fixed amount of time) for the execution to complete

Settings:

- Name of *remote command to launch* (with or without absolute or relative path)
- Type of execution and *delay* for synchronous execution

Launch a local execution during connection

Objective: Launch execution of a program, system command or executable file on the local computer during connection with remote computer. There are two types of local execution during connection:

- Asynchronous: Execution is launched and the scenario goes to the next command without waiting for the execution to complete
- Synchronous: Execution is launched and the scenario waits (during a fixed amount of time) for the execution to complete

Settings:

- Name of *command to execute* (with or without absolute or relative path)
- Type of execution and *delay* for synchronous execution

Launch a local execution without connection

Objective: Execute a program, system command, executable file (batch, script, shell, etc.) on a local computer without connecting to a remote computer.

Execute messaging agents

Objective: Execute the Lotus Notes or Microsoft Outlook or Microsoft Outlook Express messaging agents on a MediaTransfer Client. This is done via the **Launch local execution** command if the scenario is executed by the client and **the Launch remote execution** command if the scenario is executed by the server.

Send a date synchronization command

Objective: Synchronize the calling computer's date and time with the called computer's date and time.

Send a date synchronization request

Objective: Synchronize the calling computer's time with the called computer's time.

End a call

Objective: End the call without executing the remaining commands.

Execute a sub-scenario

Objective: Request the execution of another scenario from within the current scenario. In most cases, the "sub-scenario" has already been sent to the remote system.

Settings:

- Name of *sub-scenario* file (with or without absolute or relative path)
- Variable to use for generating a unique scenario name by contact on a server

Launch local execution after call

Objective: Launch the execution of a program, a system command, or an executable file on the computer executing the scenario after a call. This execution can be set with or without the condition that all call commands must have been properly executed. It can be launched in two ways:

- Asynchronous: Execution is launched and the scenario continues immediately to the next command without waiting for the execution to complete
- Synchronous: Execution is launched and the scenario waits (for a fixed amount of time) for the execution to complete

Settings:

- Name of *command to execute* (with or without absolute or relative path)
- Type of execution and *time* required for synchronous execution
- Conditions of execution (always, either if the call commands are OK, or if at least one call command is *not* OK)

Conditional Language

You can define variables (or conditions) in a scenario and then test them using the **IF**, **ELSE** and **ENDIF** commands.

Four types of conditions can be defined:

- File or folder condition (local or remote)
- Variable condition on MediaTransfer variables or system variables
- Return code condition for the last command
- On the line transfer rate (in Kb/s)

A file or folder condition can check:

- If a file or folder, or set of files and folders, is present or missing
- The amount of time that has elapsed since a file was last modified
- The date a file was last modified
- The size of a file
- The contents of a file

MediaTransfer or system variables can be compared with an entered value to see if they are equal to, greater than, greater than or equal to, less than, less than or equal to or not equal to the value.

A condition on the execution result of the last command may refer to:

- The return code condition for the last executed command (local or remote)
- The number of files transferred by the previous MediaTransfer command
- The result of the previous MediaTransfer command

MediaTransfer Variables

MediaTransfer variables let you write general scenarios that are customized upon execution depending on several factors such as caller, callee, execution date and time, etc.

Example: An identical scenario for all remote computers lets them call the server and transfer their local “sales” file to the server with the name “sales%ID_CALLER%” (where ID_CALLER is the identifier of each remote computer).
List of available variables:

ID_CALLER	: Caller identifier
COMP_CALLER	: Caller’s company name
NAME_CALLEE	: Callee name
ID_CALLEE	: Callee identifier
COMP_CALLEE	: Callee’s company name
ID_REMOTE	: Contact identifier
COMP_REMOTE	: Name of contact's company
DATE	: Date (Formats: YYYYMMDD, YYMMDD, DDMMYYYY, DDMMYY, MMDDYYYY, MMDDYY)
YEAR	: Year (AAAA)
MONTH	: Month
DAY	: Day
HOUR	: Hour
MINUTE	: Minute
SECOND	: Second
TIME	: Hour
WEEKDAY	: Day of the week
YEARDAY	: Day of the year

ERRORLEVEL : Return code of last command
LOCAL_ALIAS : A contact attribute defined and indicated on the local system.
REMOTE_ALIAS : A contact attribute defined and indicated on the remote system.
$*variable* : Environment variable defined or evaluated on the local system
#*variable* : Environment variable defined or evaluated on the remote system

You can use the DATE, DAY, WEEKDAY and YEARDAY variables to add or subtract one or several days.

Directories

MediaTransfer Server Directories

The **unique directory** lets you manage all MediaTransfer remote clients. You can record the following information in the directory for each contact:

- Identifier: name and password
- Access number (TCP/IP address, telephone number, etc.)
- Workspace allocated on the server
- Workspace access rights: read, write, execute
- Activate SSL authentication (if this contact has a certificate)
- Enter a value for the alias defined
- Name of scenario to execute when receiving a call from a client in slave mode

The directory can also include lists of contacts. Just name a list in a scenario to call all the contacts in the list.

Directory information can be **imported/refreshed from a source external** to MediaTransfer **Server** that is accessible via Microsoft ADO (relational database, LDAP directory, ODBC accessible data source).

MediaTransfer Client Directories

The directory of **Calling Servers** contains the lists of MediaTransfer remote servers that are authorized to call each MediaTransfer client. You can record the following information in the directory of Calling Servers for each server:

- Its identifier: name and password
- Access number (TCP/IP address, telephone number, etc.)
- Secondary (backup) server access number (TCP/IP address, telephone number, etc.), if installed
- Workspace allocated on the client
- Workspace access rights: read, write, execute
- Activate SSL authentication (if this contact has a certificate)
- Enter a value for the alias defined

Server and Station Scheduler

The scheduler is used to automatically launch scenarios on the server and on the clients. On the server, the scheduler is also used to automate imports into the directory of contacts.

The scheduler polls the trigger elements associated with each scenario (or import). These elements are time-based or event-based (file/tree structure contents or characteristics). It launches scenarios (or imports) when the element has occurred.

The scheduler also scans one or more queues configured by the administrator. These queues are folders where any scenario placed therein is automatically executed.

Communication Performance

Differential synchronization

Objective: optimize the volumes exchanged when using the **Synchronize files** command to **update** large files remotely (update a Web site, multimedia files, applications, etc.). Differential synchronization only transfers the differences between each source and target file, not the entire source file.

Table of measurements:

Example of file	*File size*	*Volume transferred*	*Gain*
Database	219 MB	3.6 MB	98%
Executable	287 KB	160 KB	44%
Word file	5.4 MB	1.3 MB	77%

NB: The differential synchronization can be deactivated according to the size and the type of the files.

Online compression

Objective: optimize volumes exchanged when transferring all file types. MediaTransfer uses built-in, high-performance online compression based on the LZW algorithm. Compression can be activated or deactivated in the general configuration options. When activated, compression is used transparently for all file transfers. MediaTransfer's online compression automatically does the following:

- Compresses files on the sending computer before their transfer
- Decompresses files on the receiving computer after their transfer

Table of measurements:

Measurements were taken for .exe files using an ISDN connection (64 kbit/sec). Compression levels are as efficient as those achieved using PkZip.

File size	*Duration without compression*	*Duration with compression*	*Gain*
250 Kb	33 s	10 s	70%
1 Mb	134 s	19 s	72%
10 Mb	1334 s	368 s	72%

MediaTransfer's online compression rate is as efficient as that provided by the most popular compression software on the market.

NB: The online compression can be deactivated according to the size and the type of the files.

Dynamic bandwidth management on TCP/IP

Objective:

1) To improve the way in which bandwidth is shared with other applications that need it. MediaTransfer Server can dynamically limit the transfer rate used for data exchange as soon as it detects that another application is also using the bandwidth available.
2) Allow part of the available bandwidth to be permanently reserved for other applications. MediaTransfer Server limits the transfer rate so that it does not exceed a given value.

Automatic updating of IP addresses

Objective: automatically update, in the server directory, the IP addresses dynamically attributed at each connection on client stations connecting to ADSL or GPRS type networks. Once its directory is updated, the server can call these client stations.

Security

Availability:

MediaTransfer Server version 6 provides new features to guarantee better availability of the communication system:

- In the event of a communication malfunction, MediaTransfer Server sends a warning to an administrator (TRAP SNMP)
- Communication channels can be distributed between several MediaTransfer Servers in a given network
- Interfaces with standard monitoring tools (MOM, Patrol, etc.)
- Mutual monitoring (MIB SNMP) across several MediaTransfer servers Using this configuration the servers can share the workload, as well as provide extra security if one of them should fail

Confidentiality and integrity:

Data transferred by MediaTransfer version 6 may be secured using the SSL protocol. All the data transferred between the two computers using MediaTransfer are then encrypted using the AES algorithm with a 128-bit key.

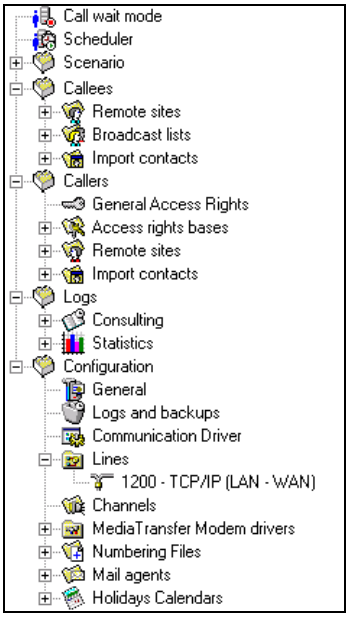
Authentication and non-rejection:

For greater security, a mechanism authenticating one or both the contacts using certificates can also be set up.

MediaTransfer Server Administration Console

User friendly interface:

- **Tree structure** (tree structure on the left and list on the right) enables the view of different entities to be structured and lets the administrator go directly where she or he wants to go.
- **Meaningful** icons.
- **Tabs** showing properties, attributes and context maintained when moving from one entity to another. This allows you to eliminate unnecessary clicks when viewing important collections.
- **Pop-up help** to get an immediate explanation of a particular point displayed on the screen.
- **Online help** for a more detailed explanation of MediaTransfer features.
- **Pop-up menus.**



Reports:

Reports are prepared in Crystal Report. You can view them on screen, print them or adapt the data before they are sent to individual recipients (attachments in HTML, CSV formats, etc.).

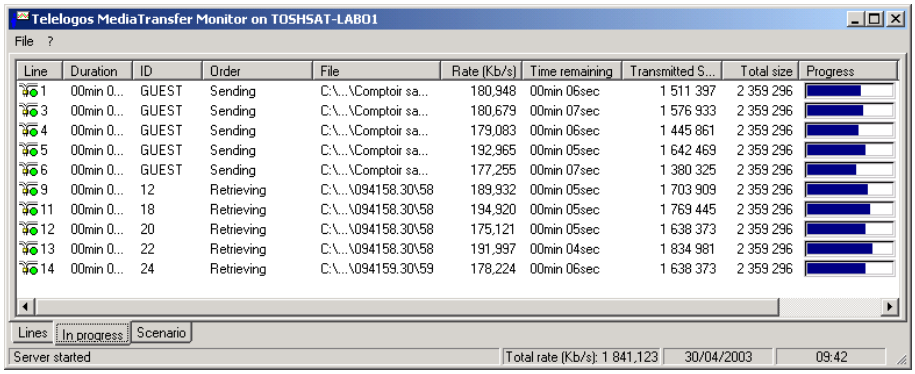
Zoom through histories, going from a general level to more detailed levels.

The console is **executable from any Windows (2003, 2000, XP) station on the network (LAN-WAN).**

MediaTransfer Server Communications Monitor

The communications monitor lets you display the details for:

- Available communication lines and their configuration
- Communication lines currently used
- Server scenarios that are being executed



MediaTransfer Client's GUI

All client stations, except for Windows CE clients, have a GUI (Graphical User Interface) that is very similar to the administration console. The GUI is frequently used only when evaluating MediaTransfer**,** since the goal of the software is to work invisibly on each remote station.

Logs and Statistics

MediaTransfer logs all events. The length of time logs are kept can be modified. Logs are used to produce statistics on the volumes *exchanged* and the processing times by hour/day/week/month/year information for every relevant level (Global, User).

MediaTransfer's administration console lets you view and print two types of statistical reports:

- Server activity in terms of the number of incoming and outgoing calls analyzed by line and contact. Use it to detect server overloads or anomalies per line and contact.
- Progress of connections for callers and callees.

MediaTransfer also lets you automate the generation of statistical files that can be used by other applications (for example: files can be used by Excel to automate printing, emailing and conversion into HTML format of generated statistics).

Download from the Web

Visit www.telelogos.com to download demo versions of MediaTransfer **Server** and **Client.** Demo versions are full versions that allow you to evaluate the software for 30 days after installing it. Our technical support department is available to guide you through the installation or parameter setting of the software. You can contact us at any time:

- By telephone: +33 (0)2 41 22 70 00
- By fax: +33 (0)2 41 22 70 22
- By Email: support@telelogos.com

Languages

MediaTransfer is available in English, French, German and Spanish.