



HOME

User Manual

DELMIA Process Engineer[®]

PoolingServer and Server Tools



Foreword

This manual provides an introduction to the basic operations and functions of the Server tools.

While developing these functions we have made every effort to create a clearly organized, easy-to-understand program structure.

A user-friendly interface as well as a clear menu guide will enable you to quickly learn how to operate the program and to get familiar with its functions so that you can carry out your planning tasks in a quick and reliable way.

No Liability or Guarantee

Our programs and manuals have been compiled with great care and to the best of our knowledge. They have also been tested in a production setting. However, we assume no liability and provide no guarantee that the software and related descriptions are free of error or are suitable for special purposes.

DELMIA assumes no liability for any damage that may arise from the use of this software. By using this software, the user acknowledges this exclusion from liability and shall hold DELMIA exempt from all claims.

Copyright

The information in our documents may be copied and distributed for internal purposes provided it is done free of charge and the contents are not altered or distorted.

Any other form of usage, especially the sale on CD-ROM or in any other publication in whole or in part is only permitted after prior written consent by DELMIA.

Some parts of this software are owned by Unigraphics Solutions Inc. and are copyrighted © 2010. All rights reserved.

Some parts of this software are owned by combit® GmbH and are copyrighted. Report-/Print module List and Label® Version 8.0: Copyright combit® GmbH 1991-2010.

Modifications

Moreover, DELMIA retains the right to make modifications and improvements to the product described in this manual at any time without prior notification.

DELMIA and the 3DS logo are registered trademarks of Dassault Systèmes or its subsidiaries, in the United States or other countries.

© 2001-2010 Dassault Systèmes - All rights reserved

Table of Contents

1. Introduction	1
1.1 How to Use this Manual	1
1.2 Documentation Conventions and Symbols	1
1.3 New Functions in Server Tools	2
2. Pooling Server	3
2.1 Multiserver Environment	3
2.1.1 Advantages of Multiserver Environment	3
2.2 Server Pooling	3
2.3 Server Processes, Clients, and Server Tools	5
2.4 PoolingServer and Setup	6
2.4.1 Installing Master Server	7
2.4.2 The PoolingServer Configuration	8
2.4.3 Load Balancing	19
2.5 Settings	21
2.5.1 PoolingServer Settings	21
2.5.2 ServerTools/ServerPool Settings	25
2.6 Failover Handling	28
2.7 Global Emergency Mode	32
3. Server Tools Client	33
4. Logging onto PoolingServer and Server Pool	35
4.1 Starting the ServerTools Client	35
4.2 Menu Bar and Toolbars	37
4.3 Working with Server Tools Client	37
4.3.1 Explore the Server Installation	38
4.3.2 Monitoring the Server Installation	39
4.3.3 Changes in R18	44
4.4 PoolingServer Context Menu	45
4.4.1 Settings	45
4.4.2 Show ServerPool Load Indexes	47
4.4.3 Lock PoolingServer	47
4.4.4 Parse PoolingServer Configuration	48
4.4.5 Terminate Unused IPD Server Processes	50

4.4.6	Terminate Processes	51
4.4.7	Terminate Clients	52
4.4.8	Reload and Show Running ServerPools	52
4.5	ServerPool Context Menu	52
4.5.1	ServerPool Settings	53
4.5.2	Show Process Information	53
4.5.3	Terminate unused IPD Server Processes	54
4.5.4	Terminate IPD Server Processes	55
4.5.5	Terminate Clients	55
4.5.6	Reload	55
4.6	Server Context Menu	55
4.7	Context Menu on the Client	57
5.	Terminating Clients	58
5.1	Procedure	58
5.1.1	Starting Clients Termination	58
6.	EPUnlockTool	62
6.1	General Information	62
6.1.1	Mode of Operation	62
6.2	Operating Manual	62
6.2.1	Release Persistent Locks through Slave Server	63
6.3	Support for Command line Interface	64
	Appendix	66
	List of Figures	81
	List of Tables	83
	Index	84

1. Introduction

This manual explains how to use the Process Engineer Server tools for your planning purposes. This user manual is primarily intended for system administrators or employees who have the same authorization.

1.1 How to Use this Manual

This manual enables you to get familiar with the operation and functions of the Process Engineer. This manual briefly describes:

- Overview of the PoolingServer and the Multiserver Environment
- Server Tools
- Termination of Clients
- Removal of Persistent Locks



Note

When handling the Server Tools functions, please also refer to the general introduction to Process Engineer in the General Introduction Manual.



Click [General Introduction](#) to access the manual.

1.2 Documentation Conventions and Symbols

The symbols used in this manual are intended to provide you with keys to the contents in an immediately understandable manner.



This symbol is used to introduce key concepts that are covered in the sections immediately following this symbol. As a result, this symbol most frequently appears at the beginning of chapters or sections.



Note

*This symbol is used to mark notes, which provide you with additional information you need to have for further work. You will either find the Note sign at the beginning of a chapter or in a particular text passage in the chapter. Texts bearing this sign are additionally marked with **Note**. The text is always in italics.*



Caution

This symbol indicates that the text that follows describes particular circumstances that you must avoid to avoid potential errors with the operation of the program or harm to data. You will either find the Caution sign at the beginning of a chapter or near a particular text passage in the chapter. Texts that are in-

roduced by this sign are additionally marked with **Caution**. The text is always in italics.

Example

This symbol marks examples which serve to illustrate a certain situation.

1)

This symbol marks the individual operational steps involved in a particular operating instruction. Operating instructions describe operational steps, for example, how to open a menu or execute a function.

■

This symbol marks listed subjects. The symbol for listed subjects can be either used to structure a continuous text or to list main subject keywords.

➤

This symbol marks list inside a bulleted or numbered list.



This symbol marks cross reference information that is available in another manual.

1.3 New Functions in Server Tools

Support for Command Line Interface

EPUnlockTool provides support for Command line interface (CLI).

2. Pooling Server

2.1 Multiserver Environment

If several users work with programs sharing the same data they need to store these data centrally, e.g. in a database. If these users want to be notified about data changes or want to lock the data against changes from other users, e.g. while working on it, they need a central program service coordinating the data access activities of all users.

The central service programs run on a server machine that the clients connect to. Today's client PCs use the services of the server, but they can also save and execute extensive applications and data. Applications which previously ran only on mainframe computers are thus divided into a serverside central part and a clientside local part and are distributed on client and server machines. Using such a client/server environment, the required computation capacity is spread over several machines and can easily be scaled up (or down) following the usage requirements.

In a multiserver environment, multiple server processes run on several server machines to provide the central services for these client/server applications.

2.1.1 Advantages of Multiserver Environment

There are various reasons for using several servers in a client-server applications. The security of the data provision, i.e. the permanent availability of data, often plays a major role in this regard. If the server fails in a client/server network with only one server (as is the case with a mainframe) none of the clients can continue to work. In order to avoid this, the service is provided simultaneously on several servers. If a server fails, clients continue to be served by other server machines. For the provision of data services with good performance the multiserver enhances the performance of the server application.

When in a multiserver environment, the use of the computer capacities (processor, memory) can be equalized over all of the server machines and the scalability for the multiuser operation and the response time for the user improve significantly. Load balancing the use of hardware is effective in reducing and isolating performance bottlenecks, since only a small part of the users share the same server process.

DPE provides a multiserver environment with the Master/Slave installation and enhances this installation with administration and configuration capabilities by the **Pooling Server** and **ServerPool** server programs to offer optimal machine capacity scaling.

2.2 Server Pooling

The term "Server Pooling" summarizes the functionality of the separate server programs PoolingServer and ServerPool.

Using Server Pooling, IPD server processes run on multiple server machines in the DPE multiserver environment in order to serve the client to a PPR server with sufficient capacity available. The machine capacity sharing is mainly controlled by the PoolingServer loadbalancing and by some IPDServer usage limitations applied by the ServerPool to prevent process overload.

The IPD server processes are available on every server machine in a server-pool. The serverpool determines the capacity usage values of the IPD server processes and transfers them to the PoolingServer to be evaluated by the load balancing.

When connecting a PPR client to a PPR server, the **PoolingServer loadbalancing selects the server machine with the lowest current load**. The **Server Pool, running on each server machine**, provides the "next available" IPD server process. This decision depends on the ServerPool configuration values "Clientlimit" and "Memorylimit", which specify usage limitations to determine whether an existing IPD server process still has the capacity for a new client connection or a new IPD server has to be started.

A more detailed explanation of the PoolingServer and Serverpool configuration can be found in the sections ": Please refer to the [The PoolingServer Configuration](#) and [ServerTools/ServerPool Settings](#).

The DPE installation consists of a master server, optional slave servers and the individual clients.

The **PoolingServer** is installed on the **Master Server** and after login of a client and based on established parameters, such as the current available capacity, assigns the client an IPD server on a selected server PC.

The slave servers enable the server installation to be adapted to the requirements of a multi-server environment.

Some of the requirements of the Multiserver environment include, for example:

- Number of clients logged on
- Number of simultaneous accesses
- Data throughput per user

A Server Pool runs on every server (slave or master) within the EPServertools process. A new client is always logged on via the PoolingServer (**red line**) and assigned to an IPDServer process via the Server Pool (**blue line**) depending on the available capacity.

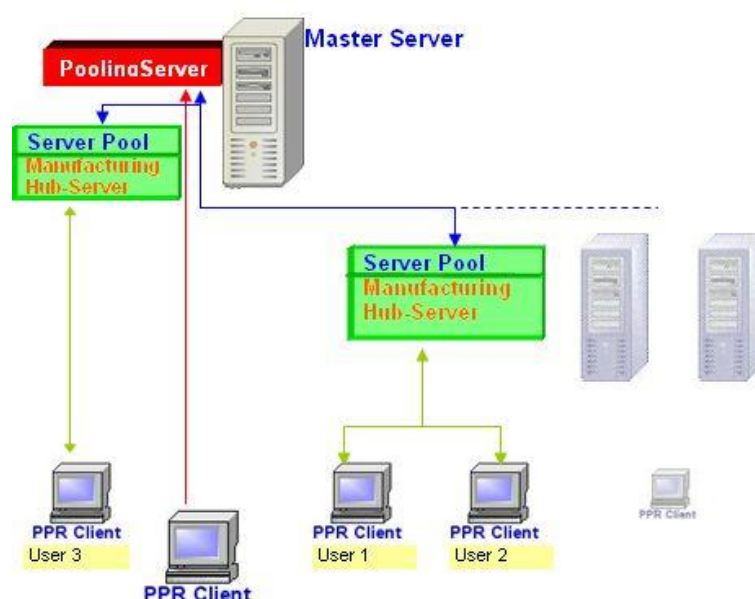


Figure 1: PoolingServer Diagram - Server Tools and Clients

2.3 Server Processes, Clients, and Server Tools

The **Server processes** provide the client with data services, administrative and functional services.

The IPD Server processes provide the client with access to the data contained in the **Integrated Process Database (IPD)** of the DELMIA Process Engineer. Data that all clients access is managed centrally in the IPD.

In addition, functional services are provided such as updating data, blocking data and checking on access rights.

The **Server Tools** monitor the IPD Server processes and manage information on the logged-on clients. The server tools are installed on master and slave servers.

In [Figure 2](#) server allocation to the client via the PoolingServer is schematically represented.

Following is a short description of the process of client log-on and assignment of a server:

- When a client logs on, he first turns to the PoolingServer of the Master Server.
- The role of the PoolingServer is to “know” the load data of all server pools (on the slave servers and/or on the master server itself), since it continuously receives utilization information from all Serverpools, such as CPU usage and memory use. Based on these load data the PoolingServer selects the machine with the most load capacity by the next client connect.
- The server with the lowest load index will be assigned to the client. The calculation of the load index can be configured and thus the assignment decision can be directly influenced. More detail regarding these control mechanisms will be provided in the following sections.

You will also find out which server processes start up and from which server machine they are executed in the sections below.

Before giving a more detailed description of the PoolingServer and the Server Tools, however, some terms that are used repeatedly in the information to follow should be explained in more detail.

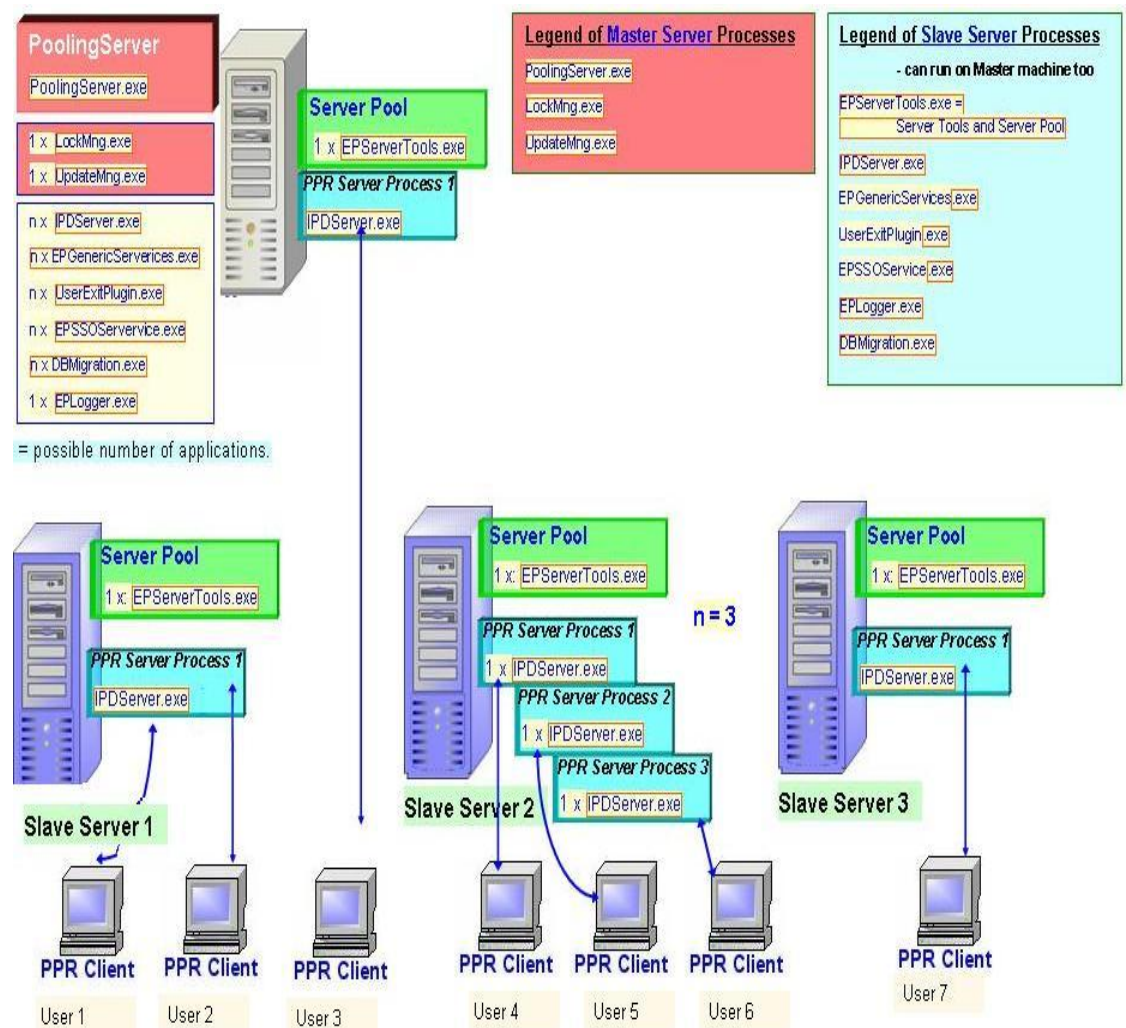


Figure 2: Which Processes Runs Where

2.4 PoolingServer and Setup

In Version PE 5.11 and later, a master-slave server installation is used, in which slave-server PCs can be added **after** installation by means of the PoolingServer configuration file (not by means of the client setup).

This add-on can be done during the run time of the Process Engineer, i.e. at any point in time.

During the Client setup the Master Server (computer name) must be specified. Since the PoolingServer runs per default on the Master Server the client will first connect the PoolingServer and then will be connected to the IPDServe via the PoolingServer loadbalancing.

- Client registry entry specifying the PoolingServer machine to connect to:

HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\ergoplan\ipdserv

What Slave machines the PoolingServer will balance is specified in the **PoolingServer configuration file**.

For some dedicated installations providing administrative services (e.g. imports) it can be useful to run a separate PoolingServer. Since the PoolingServer is delivered like any other server process in each server installation, it is possible to run an additional PoolingServer on a Slave to provide an isolated sub installation. The clients using this separate PoolingServer need to

connect to this machine. But care must be taken to not overlap the Slave pooling between several PoolingServers. Since the administration of such a sub installation complicates the overall control it should only be applied if the central PoolingServer solution on the Master has considerable drawbacks on stability or performance.

2.4.1 Installing Master Server

- 1) The installation of the Oracle database, the master server and the PPR client (if desired) is done in the same way as in Version 5.10.
- 2) After installation, the directory "...[Installationspfad]\PPRServer" must be released.
 - This will later be used as a common **data** directory of the server.

	PE 5.11	PE 5.18
Installed COM server processes	IPDServer.exe, LockMng.exe, UpdateMng.exe, RightsAdm.exe, EPServerTools.exe, EPPoolingserver.exe	LockMng.exe, UpdateMng.exe, EPServerTools.exe, EPPoolingserver.exe IPDServer.exe, EPGenericServices.exe, EPLogger.exe, UserExitPlugin, EPSSOService.exe, DBMigration.exe

Installation of Slave Server

- 3) Enter the Oracle service name.

	PE 5.11	PE 5.18
Installed COM server processes	IPDServer.exe, RightsAdm.exe, EPServerTools.exe	EPServerTools.exe, IPDServer.exe, EPGenericServices.exe, EPLogger.exe, UserExitPlugin, EPSSOService.exe, DBMigration.exe

Server Pooling process explanation:

- EPPoolingServer.exe: provides the Server Loadbalancing
- EPServerTools.exe: provides the ServerPool and the ServerTools' administrative server functions

The PPR server processes in the DCOM configuration must be configured according to the Delmia guidelines on all PPR server computers. See the section DCOM configuration of the PPR server processes.

All PPR server computers used must as of version PE5.11 be listed in the pooling server's configurations file, so that PPR clients can connect to the PPR server.

2.4.1.1 Registration Editor (for Slave Servers only)

The "LockMng", "RightsAdm", "UpdateMng" services may only run on the master server. Starting from PE 5.11 the Service "RightsAdm" are void. These registration entries cannot be made during installation and must therefore be done afterward. The entries are made under the key::

"HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\IPDSERVER".

The Master Server ist registered under: ... \UpdateMng\Cascaded\Server.

2.4.1.2 Installation of Client

- 1) Specify the master server as the server.
- 2) Select the released **data** directory of the master server.
- 3) Install Administration Tools and leave all other settings as they are.

2.4.2 The PoolingServer Configuration

The DPE multiserver environment is set up with the help of the pooling server configuration. If multiple PPR server machines are used, the PoolingServer configuration must be edited after the setup is executed in order to make possible client-server operations in multiple server computers.

The PoolingServer configuration is edited and saved in the PoolingServer configuration file.

2.4.2.1 Description of the Configuration File

The default location of the configuration file is the folder:

... \PPRServer\data\poolingsrv. In the configuration file you can find a short description to the specific definitions. You can also save the configuration file in another location. The settings you need to make are described in the following sections.

The configuration file contains all PPR server computers and optional data on the configuration of the capacity usage measurement. The server and client computers are assigned to one another with group names. The assignment makes it possible for a subset of PPR client computers to connect exclusively to the assigned subset of PPR server computers.

The configuration file can be edited during the server run time. If changes are made, it takes a maximum of 20 seconds after saving for the PoolingServer to react.

After installation, the PoolingServer configuration file should look similar to [Figure 3](#). All of the comment lines were removed in this case.

```
/** DPE POOLINGSERVER CONFIGURATION **/
```

```
<<netconfig>>
```

```
server net:
```

```
    Location [localhost(0)]
```

```
//localhost is the default master server
```

Lines that begin with
//
are comment lines

Figure 3: PoolingServer Configuration File

The PoolingServer configuration file is divided into two sections:

- 1) The section for declaring the server to the [server net](#) and
- 2) The section for declaring the client to the [client net](#).
 - If clients should only access certain servers on the network, e.g. in order to prevent WAN network connections, multiple server groups must be set up – see configuration example b).
 - If all clients should be able to access all server computers, it is not necessary to divide the computers into groups. Only one server group needs to be created – see configuration example a). Client machines must not be listed in this case.

Two possibilities are described in the following:

a) All Clients can access all servers.

```
<<netconfig>>

servernet:
    Fellbach [localhost, Server1, Server2]

clientnet:
```

b) **Client-Server subnets** are defined. This enables a certain group of clients to have access to a certain group of servers to, e.g. (*localhost* is the master server):

```
<<netconfig>>

servernet:
    Fellbach [localhost, Server1, Server2]
    Location2 [Server1, Server2]

clientnet:
    Fellbach [client1, client2, client5]
    Location2 [client3, client4, client5]
```

or if using server machine prioritization:

```
<<netconfig>>

Servernet:
    Location1 [Server1 (0), Server2 (1)]
    Location2 [Server3 (0), Server4 (0)]

Clientnet:
    Location1 [Client1, Client2]
    Location2 [Client3, Client2]
```

Comment: -[Fellbach](#), [Location2](#) are the configured server domains.

- The **Client-Server subnet** configuration allows utilization of the local network topology and can result in better data throughput. It can also be used to exclusively assign server machines to certain clients – e.g. for usage-intensive tasks such as import/export .

The following rules must be observed:

- As many server domains as desired can be configured ([Fellbach](#), [Fellbach2](#))
- Per default all server machines have the same machine priority in the load balancing. As another option, different machine priorities can be specified in parentheses, such as:
[Fellbach \[localhost\(0\), Server1\(2\), Server2\(1\)\]](#)
 - the highest priority is 0 (best PC)
 The computer priority can be the same for several computers, e.g.:
[Fellbach \[localhost\(0\), Server1\(0\), Server2\(0\)\]](#).
 By default, if no priority is specified, the highest priority 0 is used.
 Server computers can be listed in an unlimited number of server domains (groups).
 A server machine can thus be a member of one or more server domains, e.g.:
[Location1 \[Server1\]](#)

[Location2 \[Server1\]](#)

If a server is listed in multiple domains, you must ensure that the server groups correspond to the network topology. This means that if a server computer is a member of 2 domains, the network connection for clients of both domains should also be available (latency time, bandwidth).

- Client PCs may be listed in as many server domains as desired. A client can therefore be a member of one or several server domains, such as: [Location1 \[Client2\]](#)

[Location2 \[Client2\]](#)

The guideline for complying with the network topology also applies to the assignment of a client computer to several server domains, since the client could potentially connect to all server computers of the assigned domains, but the network values (latency time, bandwidth) might not be taken into adequate consideration (with the criterion server pool response time) in the workload balancing process of the pooling server.

- Client PCs do **not** necessarily have to be listed for a server domain. There are two possibilities:
 - a) If a client does not appear in any server domain, **ALL** servers are available to it from this selection, the PoolingServer assigns the Client an IPD server instance on the server PC during logon.
 - b) Clients that are not listed are automatically connected to the reserved server domains '@anonymous_userdomain', e.g.:

servernet:

[@anonymous_userdomain \[server1\]](#)

In the example, all clients which are not listed are connected to the server 'server1'.

- If a client is listed for at least one server domain (Fellbach, Client1), the server selection is limited to the servers listed in the server domain (e.g. [Fellbach \[localhost, Testserver1, Testserver2\]](#))

A minimal configuration, without listing any client PC, would be for example

```
<<netconfig>>
```

```
servernet:
```

```
    Location [localhost, Testserver1, Testserver2]
```

```
clientnet:
```

With this configuration, all clients can connect to the server computers 'localhost', 'server1', 'server2'. These connects will be load-balanced as it is done for all listed client PCs.

Machine Names

In the configuration of PoolingServer you can enter the name of machines as follows:

Alphanumeric ASCII string (0...9, A. . . Z, a ... z) without blanks and special signs. Alphanumeric ASCII string may be contained one point '.' and/or one underscore '_'.

Example

```
servernet:
```

```
    [ abc123, abc_123 , abc.def ]
```

To use ASCII special characters, the characters must be set into simple apostrophes '...'.

Example

```
servernet :
    [ 'abc-de-123', 'abc de', 'abc:def', '#123abc' ]
```

UNICODE signs outside the ASCII area aren't allowed generally in the PoolingServer configuration file.

2.4.2.2 Administration Clients

Administration clients are client computers that can connect to an IPD server even if the Pooling Server is locked for normal client connects. This capability supports administrative activities while normal users are not able to connect. Client machines for administrative purposes are declared through the assignment to the admin-domain of the PoolingServer configuration.

- The admin-domain is a reserved domain and will be listed in the section 'clientnet' of the PoolingServer configuration file; its name @admindomain cannot be changed or used as another domain identification.

Example

```
clientnet:
    @admindomain [client1, client2]
```

The computers client1 and client2 can connect to an IPDServer via the PoolingServer even if the PoolingServer runs in locked mode, e.g. by setting 'locked_on_start' = 1 or by manually setting the lock mode in the Server Tools.

2.4.2.3 Supporting DNS Notation for Server Machine Names

The PoolingServer configuration supports listing the machines names by the DNS (Domain Name Server) notation. This allows to administer the Server-Pooling across different Windows domains and to list the machines by globally unique names. The DNS machine notation (e.g. client1.deg.ds) can be applied for the servernet and the clientnet.

The DNS notation can be mixed with the simple machine notation but has a higher priority for the PoolingServer configuration since it is more precise - i.e. if the same machine is both listed with and without DNS notation the DNS entry will be considered.

Example

- clientnet:
 - Location1 [client1.deg.ds@ @dpfframe, client2@ @pprloader]
 - Location2 [client1.deg.ds, pyramid]

2.4.2.4 Extend ServerPooling to Support Administration Clients

The PoolingServer configuration file allows grouping client and server machines and assigning each other. Using this configuration a client can be routed to a certain server machine. In most cases a server machine will be configured for more than one client. Depending on the current ServerPool client limit and the number of client connects a certain number of IPDServer processes will be started and shared between the clients. In R16 the PoolingServer configuration does not allow to assign a dedicated client to a certain IPDServer process to use it exclusively, instead only the server machine can be addressed.

For some clients, especially dedicated administration clients, there is a need to connect and use an IPDServer process exclusively to be able to use the

whole process resources (memory, CPU per Thread) for memory or performance critical tasks.

The Poolingserver supports these clients with the exclusive connect API. Also, exclusive connects are additionally supported by the PoolingServer configuration using exclusive connect options. This allows to configure clients which are build with the standard connect API to use IPDServer processes exclusively too.

Additionally to the capability to reserve IPDServer processes for exclusive usage client applications have to be addressed at process level to provide a different ServerPooling for different client applications on the same machine. Of course, the specific serverpooling for clients identified by their process names can be used for normal (non-exclusive) connects too.

In order to provide a process level resolution of the PoolingServer configuration the client application will be identified by two new client labels:
- the client process label and the client module label.

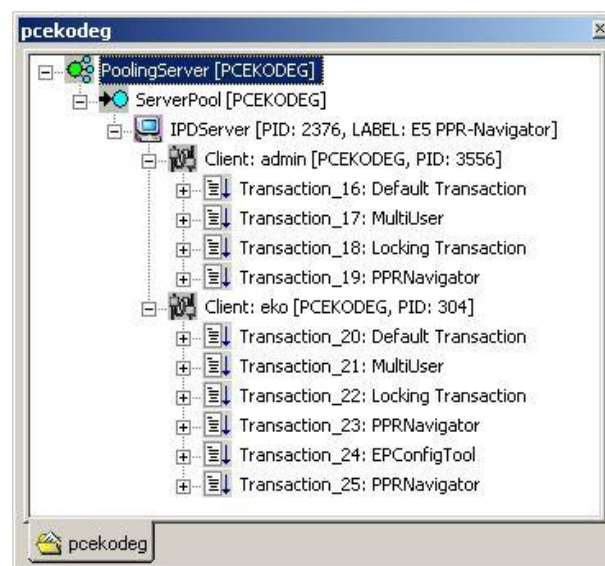


Figure 4: Client Labels

PoolingServer Configuration Details

Details of the client identification on process level:

- The first is a reserved label which identifies the client application by its exe name: `@@ <client process label>`.
- The second identifies optionally the client application type if it is set by the client programmatically: `@<client module label>`.
By that additional label different client applications can be distinguished if they run within the same client process.

These label extend the client machine name in the PoolingServer configuration.

Example for the Client Net

clientnet:

Location [client1 @ @dpfframe, client2]

// Example for a `process label`

Client1@@dpfframe and client2 belong to the client domain 'Location' being assigned to all server machines configured for 'Location'. Client1@@dpfframe

will be resolved to all client processes identified by the exe name 'dpfframe.exe'.

The PoolingServer uses that client label specification which fits most exactly to the exe/module labels. The client module label can be optionally specified for a client process during runtime by the Connect API.

If a client process label and no client module label is defined in the PoolingServer configuration for a client machine (e.g. client1 @@dpfframe for the client process dpfframe.exe), the client will be only identified by its process name even if a client would have set the module label while connecting.

a) Example for identifying the client by its process name:

```
clientnet:
  Location1[client1 @ @dpfframe]
```

If no process label is defined but a module label, all client processes are considered which set that module label.

b) Example for identifying the client by its process name and the client module label:

```
clientnet:
  Location1[client1 @ @dpfframe @label1]
```

c) Example for identifying the client by the client module label only:

```
clientnet:
  Location1[client1 @label1]
```

If the module label is the same for different client processes all these client-processes will be considered.

If both a) and b) are defined the client module label is preferred since it is considered to be most fine granular.

Whether the server is used shared or exclusively can be specified by the connect option:

```
clientnet:
  Location[client1 @ @dpfframe ( @opt = { connect_exclusive_unique } ) ]
```

If no connect option is configured the server process is shared by different clients.

The following connect options are supported in the PoolingServer configuration:

- connect_exclusive(<server exclusive label>) :
Connect to an IPDServer process exclusively labelled by <server exclusive label>
- connect_exclusive_unique:
Connects to an IPDServer process identified by a generated unique exclusive label, the IPDServer will be automatically terminated if no client is connected any more
- connect_shared:
Connects to all configured server machine being not exclusively used

The connect option connect_exclusive (<server exclusive label>) allows to connect/reconnect a sub server pool identified by the specified exclusive server label. This provides the capability of reusing exclusive labeled IPDServer Processes.

If IPDServer processes are shared by other clients connecting with the same exclusive server label they are reused considering the ServerPool client and memory limit. If one of them is exceeded a new exclusive labeled IPDServer process is started.

The connect option can also be set for a client domain. The domain setting does not overwrite a specified connect option for a client within this domain.

- Example for domain connect option

```
servernet:
  LocationAdmin [Server1]
  @anonymous_userdomain [ Server1, Server2]

clientnet:
  LocationAdmin
  [
    AdminClient1,
    AdminClient2,
    Server1 @ @pprloader(@opt{connect_exclusive_unique})
  ] @opt = { connect_exclusive('domain admin') }
```

In this example the process pprloader.exe is connected to an uniquely exclusively used IPDServer process on machine Server1. The client applications of the machines AdminClient1 and AdminClient2 is connected to Server1 on the exclusive sub server pool 'domain admin', i.e. IPDServer processes connected by AdminClient1 and AdminClient2 is shared among them while other clients will not be connected to these IPDServer processes. But other Clients can be connected to the machine Server1 too, since Server1 is listed in both domains LocationAdmin and @anonymous_userdomain.

If a client defines its connect option by using the connect API of the IEPIPDClient or IEPPoolingServer interface, it cannot be overwritten by a PoolingServer configuration option – in that case the connect option of the configuration will be ignored.

2.4.2.5 Switching off the Load Measurement of PPR Server Machines

The load measurement on the basis of Windows performance indicators can be switched off for a PPR server domain (group of PPR server machines) in the PoolingServer configuration. This change of the configuration can be made while the PoolingServer is running. After max. 20 seconds the changes become operative.

The performance load measurement should be switched off for PPR server machines on which Windows does not support performance indicators for processes or on which the system performance DLL is disabled. The availability of the Windows performance DLL can be queried in the system registry:

```
key
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\PerfProc\Performance

entry:      Disable Performance Counters

value:      0 - available, otherwise not available
```

If the value in the registration editor is not 0, the statement [disable_load_measurement](#) must be set in the PoolingServer configuration file for this machine.

Also, there can be other reasons the PDH measurement is not available on some systems or becomes unavailable at runtime. These reasons are published by Microsoft MSDN. In such cases it can be necessary to rebuild the Windows Performance indicators in the OS or to reinstall parts of the OS.

These workaround and fixes regarding the Windows PDH availability have to be maintained like any other part of the Windows OS.

Since the DPE installation cannot control the PDH availability the PoolingServer switches off automatically the PDH measurement if it becomes unavailable by Windows. In this case the load balancing will continue using an alternative load index calculation based on other load indicators.

Example from the PoolingServer configuration file:

```
<netconfig>>
servernet:
    Location [localhost(0)] @opt = { disable_load_measurement }
clientnet:
```

mation is displayed (ServerTools). The performance load indices of these machines cannot be computed anymore, instead the alternative availability load index is calculated from the following indicators:

- Number of PPR clients connected to the machine - 45% weighting
- Free available machine memory - 30% weighting
- Network response time of ServerPools - 15% weighting
- Machine priority (PoolingServer configuration) - 20% weighting
- Load increase (computed of the other indicators) - 10% weighting

The results are then displayed in the ServerTools client in brackets []. Only load indices that are based on the same computing method can be compared, that is, only indices without [] or indices with [].

The precision of this alternative load index is restricted compared with the load index calculation on PDH measurement but allows to keep the clients balanced over the server machines if the PDH measurement becomes unavailable at runtime.

2.4.2.6 Clients which are not listed in the PoolingServer configuration

To easy assign common (non-administrative) clients to a server domain without listing them all in the PoolingServer configuration the administrator can use the reserved token “@anonymous_userdomain” in the **servernet** to which all **unlisted** clients are connected.

```
PoolingServer-Konfigurationseintrag:
...
servernet:
    @anonymous_userdomain [server1, server2]
...
```

2.4.2.7 Configure the Server Emergency Behavior

The PoolingServer assigns each new started IPDServer process an own emergency configuration defined within the emergency configuration section of the PoolingServer configfile:

```
<<emergencyconfig>>
▪ config_name = @default_emergency
▪ emergency_causes
```

- database
- critical_exception
- memory
- timeouts

In the emergency section the configurable part of the IPDServer emergency behaviour can be adapted to the needs of the installation. The setup delivers two emergency configurations:

```
<<emergencyconfig>>

config_name = @default_emergency

emergency_causes:

    database:

        lock_new_transaction [ oci_error = { 18 - 602 } ]
        abort [ poet_error = { -6001 , -6061, -6082, -6102 },
                oci_error = { 12500 - 12600, 3113 - 3114 } ]

critical_exception:

    abort [ recurring_error_state = { (20, 120, type_database),
                                     (20, 120, type_c_exception),
                                     (20, 120, type_any) } ]

memory:

    abort [ process_available_virtual_kb = 10240 ]

timeouts:

    client [ abort_timeout = 300 ]
    server [ close_transaction_timeout = 900 ]
```

This default emergency configuration will be automatically applied for all connects not using a specific emergency configuration.

The setup delivers the following specific emergency configuration for all E5 client:

```
config_name = ppr_navigator_emergency
```

The only difference to the default emergency is the IPDServer memory usage limitation to prevent out-of-memory errors:

```
memory:

    abort [ process_available_virtual_kb = 102400 ]
```

These configurations delivered by the setup can be used as prototype for own definitions bound to a category of client connects (if the clients are distinguished to be connected to different exclusive pools). If changing emergency specifications using an own emergency configuration all categories of the default emergency must be listed even if some parts will be equal in that own configuration.

For detailed information about the emergency and failover handling of the server. *Please refer to the* [‘Failover Handling’](#)

2.4.2.8 Changes in the PoolingServer Configuration after PE 5.17

As of the version PE5.17 the automatic prioritization of machines in the PoolingServer configuration has been changed:

Before PE 5.17:

If no priority were specified for a machine in the PoolingServer configuration, the priority was automatically set in the order of the machine listing within a single domain. To disable the automatic prioritization the customer had to manually set each machine priority.

Change since PE 5.17:

The automatic prioritisation has been replaced by internally applying each machine with the highest priority (value: 0) unless the priority is manually set in the configuration file.

Further Extensions in R18:

- 1) The PoolingServer configuration has been extended to support the emergency configuration for Oracle errorcodes received by the IPDServer OCI layer:
 - The emergency causes for database errors can be detailed for OCI errors
 - a) non-aborting emergency by 'lock_new_transaction' and
 - b) aborting emergency by 'abort' - like for POET errors;
 - Additional support of number ranges for POET and OCI errors is included, e.g.:

```
emergency_causes:
  database:
    lock_new_transaction [ oci_error = { 18 - 602 } ]
    abort [ poet_error = { -6001 , -6061, -6082, -6102 },
           oci_error = { 12500 - 12600, 3113 - 3114 } ]
```

- 2) A new configuration options offers the possibility to allow empty domain assignments in the PoolingServer configuration. This extension was required to better support semi-automatic PoolingServer configuration changes done by external scripts or programs to adapt the PoolingServer configuration, for instance on temporary installation changes (e.g. available server machines).

```
<<netconfig>>
  @opt = { allow_empty_domain_assignment }
```

How does it work?

If the option 'allow_empty_domain_assignment' is not set (setup default) no client can connect to any server if a single server domain, for which a non-empty client assignment is defined, is empty. Example:

```
<<netconfig>>
  servernet:
    serverdomain_1 [ ]
  clientnet:
    serverdomain_1 [client1]
```

With the option 'allow_empty_domain_assignment' enabled: If the client, being assigned to the empty domain, connects it receives the error message

"No server pool assigned to this client in the PoolingServer configuration."

Other clients assigned to non-empty domains can connect without any error message simultaneously using the same PoolingServer configuration.

Additionally, empty domain definitions are removed in any case from the configuration internally if no client assignment is defined. This removal is proceeded even if the option 'allow_empty_domain_assignment' is not enabled. For example the following configuration:

```
<<netconfig>>
servernet:
    serverdomain_1 [ ]
    serverdomain_2 [ server2 ]
clientnet:
    serverdomain_1 [ ]
```

is internally replaced by removing the empty domains like follows:

```
<<netconfig>>
servernet:
    serverdomain_2 [ server2 ]
clientnet:
```

2.4.2.9 Changing the Configuration File while the Server is in Operation

The PoolingServer configuration file can be changed at any time even if the PoolingServer is already running. It will be read in each 20 seconds and, if successful, updated in the PoolingServer.

If the configuration file is changed when the PoolingServer is running and if this change is faulty (e.g. syntax error, inexistent computer name, etc.), the last error-free configuration is used. All configuration parser errors are written like all any other errors into the PoolingServer log file. After correcting the faulty configuration a succeeded log entry will be written. If the error is not rectified, entries referring to errors will continually be written to the log file. The PoolingServer configuration is re-read every 5 seconds whenever an error occurs. The log in the log folder should therefore always be checked after the configuration is changed.

PoolingServer configuration read in:

- The PoolingServer checks the configuration for changes every 20 seconds. When the configuration is re-read, the new configuration is compared to the previously used configuration and any changes made are applied in the running operation.
- In case of a faulty change to the configuration file, the PoolingServer continues to run with the last valid configuration. It checks for file changes every 5 seconds till the error is corrected.

If the PoolingServer configuration is changed before the PoolingServer is started and there is an error, the PoolingServer terminates immediately after starting. A message with the number of the erroneous line of the configuration file and the invalid token is written into the PoolingServer log file. The PoolingServer can continue to run after being started only with a valid configuration. Therefore the configuration file must first be free of errors.

A configuration can be checked in R18 at any time in parallel while another configuration is active. The new functionality is offered by the ServerTools client. *Please refer to the [Working with Server Tools Client](#).*

The PoolingServer is automatically started when the first Client connects.

- If the PoolingServer starts up, the system will check to see if at least one Serverpool is available.
- If the PoolingServer does not start up, the PoolingServer will be terminated again and the client receives an error message.

2.4.3 Load Balancing

The Poolingserver offers the capability to balance the loads between the server machines by directing new client connects to the server machine with the lowest load.

Since this capability had limitations regarding the precision for short time load changes the balancing algorithm has been extended in R18 to be able to balance frequent connect requests too (within 1 second instead of 20 seconds).

Additionally, the loadbalancing calculation has been changed in order to consider latent server memory loads, the load impact of the thread usage of large multiuser installations, to reflect the non-linear behaviour in high load scenarios and more.

Changes in detail:

- **Load index Calculation**

- a) **Latent Memory Loads**

The ipdserver processes with low workingset usage indicate a future use (since there will be loaded up to the memory limit in the future) which can have a large impact on the machine resources if this happens to nearly the same time. To consider that future impact a new coefficient “memory usage forecast” is created. This new latent memory load coefficient considers the amount of inactive pool mem (process limit 3GB - current workingset) weighted with the number of client connections of the pool. The client connection weighting take into account the inactive memory will as more be effective for the memory load as more clients are connected to the pool.

- b) **Load Impact of the Number of Threads**

An extensive amount of threads can have two extremes:

- 1) A high thread count within a single process.

- 2) A high thread count overall processes. Since these two cases will have a different impact on the multiuser behaviour even two new coefficients are created: “pool thread count” (overall processes) and “thread count” (maximum number of threads of a single process)

- c) **Correction of the Memory Usage Coefficient**

Before R18 the memory usage coefficient has been calculated from the virtual memory size of the process with the highest load index. This approach did not consider the memory consumption overall processes in the Pool. In the past it was an limitation due to the missing IPDServer process GarbageCollector (introduced in R14). Now in R18 the memory usage coefficient is calculated from the virtual memory consumption of all IPDServer processes of the pool.

- d) **Improvement of the Memorylimit Convergence Coefficient**

It considers now the average virtual memory value overall processes to calculate the difference to the pool memory limit. Before R18 only the process with the highest load index was used.

- e) **Solve Short Time Load Index Calculation Problem**

Before R18: to ensure medium and long time stability of the load index and the load trend calculation the load index was calculated only after collecting all load data within the complete measurement interval (default timespan 20 seconds).

Since it is not possible to decrease the measurement interval much without losing the load calculation stability the load index is now interpolated to provide short time index refreshes. The interpolation uses the partial load measurement together with a trend calculation. The loadindex is interpolated every 1 second on base of the current pool measurement.

With this approach the load index is updated 20 times faster than before R18 which allows to balance even lots of client connects within a short time using a one-second-resolution.

f) Improve load Index Calculation for Critical High Load Scenarios

Before R18 the load index has been calculated linearly, i.e. the load index has increased proportional with the increasing load metric, e.g. like the CPU utilization. From the experience in the past we know this linear calculation leads only to a fair balancing as long the machine load does not exceeds the middle range. For high machine loads we have additionally to deal with inefficiencies of the OS, bottlenecks to global shared machine resources and with saturation effects of the machine resources. This means in the high load range the load metric needs to be considered exponentially. The following exponential high load criteria are introduced in R18:

- f.1) CPU utilization above 35 % by square function, below 35% the machine ratio is linear.
- f.2) Exponential machine free memory calculation for memory lower than 1/2 Windows virtual process limit (== 1500 MB expected maximum workingset) by square function.
- f.3) Calculating square load gradient index instead of a linear gradient - high load changes will change the load index more effective with the next load index computation cycle (gradient loopback).
- f.4) The memory limit convergence is exponential by square function.

▪ Alternative Load Index

The calculation of the alternative 'availability' load index was changed to additionally consider the free available machine memory. This alternative load index is used if the PDH (Windows Performance Data Helper) measurement is not available at the server machine.

The additional free memory part allows to balance the current memory consumption between the machines. Since a machine memory exhaust has a critical impact on the server performance and availability (memory swapping) this is a great improvement if the PDH measurement cannot be used on a server machine. Furthermore the calculation can be adapted now by registry. Please refer to the [Show ServerPool Load Indexes](#).

2.4.3.1 Setting Changes in R18

Detailed information can be found in chapter [PoolingServer Settings](#).

Update Registry-Values

- memorylimitconvergence_weight
- memoryusage_weight
- load_distance_threshold

2.4.3.2 New Entries

- memoryusage_forecast_weight

- pool_threadcount_weight
- loadbalancing\threadcount_weight
- availability_clientcount_weight
- availability_freememory_weight
- availability_loadgradient_weight
- availability_machinepriority_weight
- availability_serverpoolresponse_weight

2.5 Settings

There are several registry settings to customize the ServerPooling behaviour for the current DPE installation. If these settings are changed using the ServerTools-Client they will be effective without restarting all DPE processes. Please refer to the [ServerTools/ServerPool Settings](#).

2.5.1 PoolingServer Settings

With the loadbalancing Settings the computation of the load index can be adapted to consider the impact of the hardware equipment to the load comparison. In standard installations with nearly the same hardware in all server machines there will be no need to change the composition of the load index, rather the default calculation installed with the setup should be used.

The **weight** criteria represent those single parts of the load index computed by weighting the measured different indicators in percent. They can be changed using the Windows Registration Editor – Registration keys as explained below.

2.5.1.1 Loadbalancing Settings

In “HKEY_LOCAL_MACHINE\SOFTWARE\”

“Delmia\Ergoplan PoolingServer\loadbalancing”

- **cpuload_weight [%]**
CPU usage of the entire machine
- **freememory_weight [%]**
Free available virtual memory of the machine
- **Machinepriority_weight [%]**
Machine priority applied in the PoolingServer configuration
- **Loadgradient_weight [%]**
Increase or decrease of the load index compared to the previously measured load index
- **Memorypagefault_weight [%]**
Memory page fault per second. Indicates the rate of workingset memory access misses.
- **Memorylimitconvergence_weight [%]**
Ratio of the average virtual memory of all IPDServer processes to the configured limit of the virtual memory for further client connects (ServerPool memory limit)

Memoryusage_weight [%]

Ratio of the virtual memory of all IPDServer processes to the free available machine memory

- **Serverpoolresponse_weight [%]**

Reverse ratio of the Serverpool network ping answers to the expected answers within the measurement interval.

Using the following settings, the interval of the measurement period can be modified:

- **serverpoolresponse_interval [sec]**

Answer cycle of the Serverpool to provide a single measurement

- **measurement_interval [sec]**

Measurement interval to compute the performance data of a server machine

- **integration_interval [measure-cycles]**

Integration interval to compute the load index from the measured data collected within a single measurement cycle (configured by measurement_interval).

- **loadgradient_interval [integ-cycles]**

Interval to calculate the rate of increase or decrease of the load index. The load gradient calculation is based on the load index calculation - therefore this interval must be significantly higher than the integration interval.

The smaller the intervals of the measurement period, the more short-term fluctuations of usage data of the ServerPool are included. And vice versa: the greater a measurement interval, the more the long-term usage values are included in the calculation of the usage index – short-term fluctuations are therefore not reflected as much.

- **load_distance_threshold**

The registry entry 'load_distance_threshold' allows load balancing strategy based on the number of connected clients instead of comparing the load index if the load index difference between pool machines is small. Since there can be static load differences depending on background services and OS activities between different machines resulting in low measured load differences this switch of the comparison method improves the balancing considerably.

- With the value '0' applied the loadbalancing behaves as before DPE5.16.
- With applying the default value '4' client connects are balanced based on the measured machine loads if the load index difference between the server machines is greater than 4. If the load difference is less than 4, the client connect is balanced on the basis of number of client connections to the server machines, i.e.. the client will be connected to the server machine with the lowest number of connected clients.
- If the value will set to '100' (sum in the registry configured load portions) or set bigger, the load balancing is accomplished only on the basis of the number of connected clients.

2.5.1.2 Loadbalancing Settings – Availability Index

The availability index is used as an alternative load index if the load measurement by the Windows PDH (Performance Data Helper) module is not available on a server machine. To support the availability index by configura-

ble coefficients optional registry entries are introduced in R18. These registry entries are not entered by the setup but can be applied if the availability index calculation should be changed.

- **availability_clientcount_weight [%]**
- **availability_freememory_weight [%]**
- **availability_loadgradient_weight [%]**
- **availability_machinepriority_weight [%]**
- **availability_serverpoolresponse_weight**

2.5.1.3 Connect Administration

- **cleanup_interval_ipdserver**
The interval for terminating unused IPDServer processes, in minutes. '0' switches the function off.
- **client_connect_timeout**
A connect timeout prevents blocking other client connects if a single connect hangs while initializing a new IPDServer process in the ServerPool. Such a connect blocking over a longer time span occurs also if the initialization of a new created IPDServer process takes too much time caused by a lack of resources under high load conditions.
- **locked_on_start**
If the registration editor entry *locked_on_start* is set to "1", the Poolingserver starts in locked mode, that is, no clients may connect to the PPR-Server (except for administrator clients).

```
key:         HKEY_LOCAL_MACHINE\SOFTWARE\Delmia\Ergoplan PoolingServer
entry:  locked_on_start
value:  0
```

As standard the value is set to "0".

The value can also be set by a batch run of the ServerTools client:

```
- locked_on_start =1: dpfframe.exe
<dpfres>epservertoolsres.dll</dpfres> /poolingserver_lock_on_start
- locked_on_start = 0: dpfframe.exe
<dpfres>epservertoolsres.dll</dpfres> /poolingserver_unlock_on_start
```

Similarly, a running PoolingServer can be set to the locked condition:

```
- locked: dpfframe.exe <dpfres>epservertoolsres.dll</dpfres>
/poolingserver_lock
- unlocked: dpfframe.exe <dpfres>epservertoolsres.dll</dpfres>
/poolingserver_unlock
```

2.5.1.4 The Setup Defaults

-key: SOFTWARE\DELMIA\ERGOPlan PoolingServer\

client_connect_timeout	180
cleanup_interval_ipdserver	5
max_timeout_failcount	30
pool_async_call_timeout	20
MinidumpType	normal
ErrorLogTraceFilter	

key: SOFTWARE\DELMIA\ERGOPlan PoolingServer\loadbalancing\

cpuload_weight	20
freememory_weight	15
machinepriority_weight	10
memorylimitconvergence_weight	8
memorypagefault_weight	15
memoryusage_weight	8
threadcount_weight	4
pool_threadcount_weight	4
serverpoolresponse_weight	10
loadgradient_weight	10
integration_intervall	6
load_distance_threshold	4
loadgradient_interval	10
measurement_intervall	20
serverpoolresponse_intervall	5
availability_clientcount_weight	45
availability_freememory_weight	30
availability_loadgradient_weight	10
availability_machinepriority_weight	20
availability_serverpoolresponse_weight	15

2.5.1.5 Registry Changes of Load Balancing:

Update Registry-Values:

HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\ERGOPlan Pooling-Server\loadbalancing\memorylimitconvergence_weight = '8'

HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\ERGOPlan Pooling-Server\loadbalancing\memoryusage_weight = '8'

HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\ERGOPlan Pooling-Server\loadbalancing\load_distance_threshold = '4'

New Entries

- HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\ERGOPlan Pooling-Server\loadbalancing\memoryusage_forcast_weight = '6'

➤ This coefficient considers the amount of inactive pool memory (process limit 3GB - current workingset) weighted with the number of client connections of the pool to consider latent memory loads.

HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\ERGOPlan Pooling-Server\loadbalancing\pool_threadcount_weight = '4'

➤ Considers the thread count overall IPDServer processes of the Server Pool.

- HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\ERGOPlan Pooling-Server\loadbalancing\threadcount_weight = '4'

- Considers the maximum thread of a single IPDServer process of the Server Pool.
- The following optional settings can be used to change the calculation of the alternative load index. These settings are not entered in the registry by the setup:
 - HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\ERGOPlan PoolingServer\loadbalancing\availability_clientcount_weight = '45'
- number of connected clients
 - HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\ERGOPlan PoolingServer\loadbalancing\availability_freememory_weight = '30'
- amount of free available machine memory
 - HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\ERGOPlan PoolingServer\loadbalancing\availability_loadgradient_weight = '10'
- load gradient
 - HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\ERGOPlan PoolingServer\loadbalancing\availability_machinepriority_weight = '20'
- machine priority
 - HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\ERGOPlan PoolingServer\loadbalancing\availability_serverpoolresponse_weight = '15'
- number of serverpool ping responses within the measure time

2.5.1.6 Logging of Setting Changes

If registry settings are changed after the installing by the setup these changes will be protocolled in the PoolingServer.log and ServerTools.log file.

- Example for the PoolingServer start entry with changed setting 'client_connect_timeout':
- 21.May.2007 15:43:19 EPPoolingServer:
- Service started < Build: PE 5.18 2007-04-18 09:14 1301 5.18.0.122 PE 5.18 Server(PE 5.17 SP2) Client(PE 5.17 SP2) > <Log severity: 1 (Information/Warning (1 - 'Warn'))> <VirtAddrSize: 2047 MB> <client_connect_timeout 120>

2.5.2 ServerTools/ServerPool Settings

- pdh_measurement_interval [sec]

As of version PE 5.14 you will find a new registration editor entry in the ServerPool settings:

Key:	HKEY_LOCAL_MACHINE\SOFTWARE\Delmia\Ergoplan EPSTools
entry:	pdh_measurement_interval [sec]
value:	'1'

The ServerPool registry entry *pdh_measurement_interval* determines the time interval at which the performance data of the machine and the PPServer processes are queried. The minimum is 1 second. The value may be increased to decrease the CPU load of the EPSTools.exe since the PDH measurement can cause a significant background load on some machines. The entry *pdh_measurement_interval* influences the update of the process in-

formation in the ServerTools client and the machine load computing of the PoolingServers. If the value used is too high (e. g. 20 seconds), the load index cannot be determined with the required accuracy by the PoolingServer (dependent on LoadBalancing configuration).

➤ Normal range is: 1...5 s

- **ipdserver_availability_check_timeout**

Key HKEY_LOCAL_MACHINE \ SOFTWARE \ Delmia \ Ergoplan EPServerTools \ serverpool
entry: ipdserver_availability_check_timeout
value: '30'

The value indicates the timeout in [Seconds] for the test of the IPD - Server – availability.

- **ipdserver_creation_timeout**

key: HKEY_LOCAL_MACHINE \ SOFTWARE \ Delmia \ Ergoplan EPServerTools \ serverpool
entry: **ipdserver_creation_timeout**
value: '120'

The server pool entry 'ipdserver_creation_timeout' indicates the timeout in [Seconds] for producing a new IPDServer process in the server pool.

- **client_connect_timeout**

key: HKEY_LOCAL_MACHINE \ SOFTWARE \ Delmia \ Ergoplan EPServerTools \ serverpool
entry: **client_connect_timeout**
value: '180'

The server pool entry 'ipdserver_creation_timeout' indicates the timeout in [Seconds] for producing a new client process in the server pool.

2.5.2.1 IPDServer Process Usage Limitations

- **serverpool/clientlimit**

Using the *serverpool/clientlimit* entry you can determine from which number of clients onwards a new server is started. The 2 GB operating system limit applies to this new server as well.

- **serverpool/memorylimit**

Using the *serverpool/memorylimit* entry you can determine from which virtual memory utilization level onwards a new server is started.

The server process usage limitations are explained using some examples.

Clientlimit:

- In **example 1**, a machine with 4 GB (4,000 MB) of memory is available.
- The default setting for *serverpool/clientlimit* = 5, and it remains unchanged.
- The daily work scenario looks as follows: 10 clients log on to the server at the start of the working day.
- The Windows virtual memory limitation per process is 2GB.

Every single client uses process memory of the server in the course of the day as its data objects are administered by the server. Two servers are started with the default setting of 5 and the total utilizable process memory is 2GB. By his experience the administrator has found that one IPDServer process can serve 5 clients as long as they are connected without running into out-of-memory problems. Since the Windows OS limits the memory usage per process (2GB), that *clientlimit* prevents in advance overusing the OS server process memory limitation.

If another client then logs onto the Server Pool, a new server process with additional 2GB process memory is assigned to him. This guarantees that sufficient process memory is available for this client and the the next.

If the amount of process memory utilized exceeds the physical RAM memory, the operating system takes over the necessary transfer to external storage.

It is recommended to have sufficient RAM installed on each server machine to run all expected server processes (parallel client connects / client limit) within the RAM machine memory. If the RAM is too low the Windows OS will swap the memory to the harddisk. This memory swapping will impact the response time behavior of the IPDServer as more as more often the memory must be swapped for a client request to the server. The available real memory (RAM) can be viewed in the task manager and should always be significantly greater than 0 for best server performance.

Realer Speicher (KB)	
Insgesamt	1048032
Verfügbar	592316
Systemcache	363656

In the next example the *memorylimit* is described.

- In **example 2** a machine with 1 GB (1,000 MB) of memory is available.

The default setting for *serverpool/memorylimit* = 1500, and remains unchanged.

If a single client uses a large part of the IPDServer process memory the default ***memorylimit*** of 1,500 MB ensures that the server process remains operable by assigning a new client a new IPDServer process, even if the *clientlimit* is still not exceeded.

However, the ***memorylimit*** should always be below the operating system limit of 2GB and should not be set too low. A limit that is set too low causes too many server processes to start up, which in the overall scenario again uses more memory, since each server process requires administrative data and creates data caches. If the memory limit is set too high, i.e. too close to the OS process memory limitation, a client request could fail by an out-of-memory error with the risk of losing the last data changes if the server cannot save the transaction anymore.

The maximum number of server processes to be expected should not exceed 4 per 2GB of RAM. The exact number can be determined from the memory occupancy of the IPD server process just after starting. The total starting usage of memory pages (not virtual size), across the maximum number of server processes to be expected, should not exceed 30-40% of the available RAM memory of the machine at system startup. Nevertheless, the RAM memoryoverusage should be avoided as much as possible too guarantee a good server performance. The best server performance will be reached if the machine does not swap process memory – this is indicated by the Windows task managers system performance information, especially by the free available memory. See also the remarks regarding memory swapping above for the client limit.

When you exit the server pools settings by clicking OK, all entries will be saved and used for all running processes as well.

- **MemoryMonitorThreshold**

- MemoryMonitorIntervalSec

The Memory monitor logs the current value of the free available machine memory as warning if the configured threshold is underrun. Each next 10 % change is logged again.

Can be configured by registry optionally:

- HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\Ergoplan EPServerTools\MemoryMonitorIntervalSec = '10'
- If MemoryMonitorIntervalSec < 0 the monitoring is disabled.
- HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\Ergoplan EPServerTools\MemoryMonitorTreshold_FreeAvailKB= '51200'
- If the free available machine memory falls below this threshold changes of the free available machine memory will be written to the ServerTool.log file.

2.5.2.2 ServerPool Settings Setup Defaults

- key: SOFTWARE\DELMIA\Ergoplan EPServerTools
 - pdh_init_timeout 60
 - pdh_measurement_interval 1
 - MemoryMonitorIntervalSec 10
 - MemoryMonitorTreshold_FreeAvailKB 51200
- key: SOFTWARE\DELMIA\Ergoplan EPServerTools\serverpool
 - cleanup_interval_ipdserver 0
 - clientlimit 5
 - ipdserver_availability_check_timeout 30
 - ipdserver_creation_timeout 120
 - memorylimit 1500

2.6 Failover Handling

Even if the server installation is administrated well failures of the server functionality, server resource problems (OS resources like handles and memory) or network connection problems can cause server instability leading to hanging or failing client requests. Since the possible number of causes and error combinations are fairly high a complete protection against a server breakdown is not possible. With the DPE Failover handling Delmia pursues the target to limit the error escalation by immediately restricting the affected server functionality and to save and close still connected clients and in the end to shut-down the affected server processes.

The Failover handling can be customized based on certain failover categories in the PoolingServer configuration, section <<emergency_config>>.

The default emergency configuration is used for all IPDServer processes if no other emergency configuration is applied for a server connect in the PoolingServer configuration.

<<emergencyconfig>>


```

config_name = @default_emergency

emergency_causes:
  database:
    lock_new_transaction [ oci_error = { 18 - 602 } ]
    abort [ poet_error = { -6001 , -6061, -6082 },
             oci_error = { 12500 - 12600, 3113 - 3114 } ]

    critical_exception:
      abort [ recurring_error_state = { (20, 120, type_database),
                                         (20, 120, type_c_exception), (20, 120, type_any) } ]

    memory:
      abort [ process_available_virtual_kb = 10240 ]

timeouts:
  client [ abort timeout = 600 ]           // second
  server [ close_transaction_timeout = 900 ] // second

```

- **emergency_causes**
specifies the error categories triggering the server emergency handling
- **database**
in this section database errorcodes will be assigned to emergency handling actions
- **critical_exception**
in this section critical server exceptions will be assigned to emergency handling actions
- **memory**
in this section memory indicators will be assigned to emergency handling actions
- **lock_new_transaction**
- emergency handling action:
defines the error states for those creating new transactions will be locked in the server
- **abort**
- emergency handling action:
defines the error states for those the emergency failover handling will be started to save and close all connected clients and to shutdown the server.
- **oci_error**
Lists all OCI (Oracle Call Interface) errorcodes considered for the specified emergency action
- **poet_error**
Lists all POET errorcodes considered for the specified emergency action
- **recurring_error_state**
number of equal critical errorstates of the specified exception type which will trigger the emergency action
- **process_available_virtual_kb**
amount of available virtual process memory triggering the memory emergency action if falling below that limitation
- **timeouts**
specifies the timeouts for saving transactions in the emergency failover handling
- **abort_timeout**
After this timeout the client emergency failover message boxes will be

automatically closed. That value should be lower than `close_transaction_timeout` to be able to consider the client users choice. If you increase this value, the users have more time to decide whether they should commit or abort their data. If you decrease this value together with `close_transaction_timeout`, the resources like memory and handles consumed by the server in emergency mode are faster returned to the operating system.

- **close_transaction_timeout**

After this timeout the server will commit all still open 'Choice' transactions. This value should always be greater than the client `abort_timeout` to be able to consider the client users choice.

Server Behavior

The server distinguishes between critical failures requiring an immediate shut-down of both client and server in an emergency case and severe failures requiring only a restricted server usage.

a) If in the server a critical emergency failure occurs the emergency mode is set in the XEPTransactionMng. At this point of time all transactions, which are in failover mode 'Abort' are aborted. All transactions which are in failover mode 'Commit' are committed. All client transaction being registered with the failover mode 'Choice' will be frozen. If the server failover timeout is exceeded these frozen transactions will be committed if they are still not closed by the client user. The server failover timeout is set to 15 minutes by default and can be configured.

b) If a severe emergency failure occurs only new client transactions and connections will be blocked. The failover procedure to close transactions is not applied. If a client tries to open a new transaction it will receive an error message indicating the current emergency state.

In either emergency case new clients will only be connected to server processes not running in any emergency mode.

a) Critical emergency failures:

- database exception, POET error codes -6001, -6061, -6082
- PRC connection error

b) Severe emergency failures:

- runtime check exception (access violation, stack overflow, invalid instruction, ...)
- all other database exceptions

Timeout

When the server enters the critical emergency mode, a timeout task is started for all transactions in 'Choice' mode to close them. This server failover timeout is configurable. Within this timeframe the server return the errorcode

`E_SERVER_EMERGENCY_HANDLER` if the client tries to proceed any action on a transaction. After the failover timeout is elapsed, all open transactions are closed and deregistered in the IPD-Server for all failover modes. After deregistration of all client transactions no client will be able to execute an IPD-Server data access method - instead a client disconnect error will be returned to the client. If the emergency mode 'Choice' is not applied on any of the transactions of the client process, the client transactions are aborted and deregistered immediately without any wait time.

To allow to automatically close the client on a critical server emergency failure, a clientside failover timeout can be configured forcing all emergency failover message boxes to close after the timeout is exceeded. This will lead to close

the client application either, since it is the only end state of the client side failover emergency handling.

Client Behavior

If the IPDServer enters the critical emergency mode, the call of any server API using the XEPTransactionMng module of the server will lead to specific emergency error code `E_SERVER_EMERGENCY_HANDLER`. It protects all functionality accessing and changing data. The client will receive this error code and will treat it in the XEPError module e.g. via method `HandleComError`. This module will handle it by checking the failover transaction registration via a new IPDServer Emergency Agent interface, if there are still transactions in mode 'Choice'. If this is the case, the client user can specify in a dialog whether the transactions should be committed or aborted. The choice of the user will affect all 'Choice' transactions of client application. The following dialog shows the success of the commit or abort. Confirming it will exit the client application by starting a termination sequence depending whether the EIPDClient enables 'ExitOnError' set by the API method `IEIPDClient::ExitOnAbortErrorState()`.

If there are no 'Choice' transactions, a message will be shown informing the user that the connected server runs in emergency mode and the client has been disconnected. Confirming it will terminate the client if 'ExitOnError' is enabled. If the EIPDClient does not enable 'ExitOnError', it returns the emergency error code `E_SERVER_EMERGENCY_HANDLER` to the caller application.

The essential change compared to previous releases is that the client will not be able to progress to any other action after receiving the server emergency error code `E_SERVER_EMERGENCY_HANDLER` since it will be specially handled by the EPEError module. This new introduced emergency handling is applied in critical emergency cases, like a database breakdown, and forces the client to close. Since it is possible to connect several IPDServer processes by a single client process using the EIPDClient dll, the emergency handling is even applied if one of the connected IPDServer processes has been entered the emergency mode.

Client applications which have 'ExitOnError' disabled in previous, e.g. DPM, will receive the errorcode `E_SERVER_EMERGENCY_HANDLER`. These clients use the EIPDClient connection check method `IEIPDClientConnection::GetServerConnectionState()`. In the emergency case it returns the check code '3'. The client should reconnect to another IPDServer process in this case.

EmergencyGuard

Additionally, in the client, there runs an EmergencyGuard in a second thread. It calls in regular times with a period of 10 seconds the IPDServer EmergencyAgent interface to get information about the emergency state. If the server is in emergency mode, a message is shown to the user the server is running in the emergency mode and that he should restart the client application. The message is displayed independently and modeless with regard to the client user interface, i.e. the user can work as usual without closing that informative message.

This additional handler is applied on the PPR client application to ensure the emergency notification on the client since there is no guarantee for returning the current server call within a timeframe if the server has been entered the emergency state. The message will not appear in the DPM client. DPM checks the emergency state of the server connection by an own handler querying the EIPDClient interface method `IEIPDClient::GetServerConnectionState()`.

2.7 Global Emergency Mode

Global Emergency Mode is raised when a master process runs into a critical error state that affects functionality used by other processes. Critical errors are then resolved automatically, such that Process Engineer does not require a manual restart.

Global Emergency Mode saves the data consistency of pending data changes if one of the Master processes experiences a critical error state impacting central service functionality.

The following new Emergency state information is introduced with Global Emergency Mode functionality:

- Low memory of UpdateMng process on Master server
- Low memory of LockMng process on Master server
- Critical exception in UpdateMng process on Master Server
- Critical exception in LockMng process on Master Server
- Too many threads started in UpdateMng process on Master Server
- Too many threads started in LockMng process on Master Server
- Too many handles allocated in UpdateMng process on Master Server
- Too many handles allocated in LockMng process on Master Server

3. Server Tools Client

The server tools form an administration console for monitoring and configuring the multiserver installation. This console is called the ServerTools Client. The Server Tools functionality is captured by the server process **EPServerTools.exe**

Using the **Server Tools** Client you can monitor IPD server processes or you can obtain information about the linked clients. IPD processes are processes running on the DELMIA Process Engineer Integrated **P**rocess **D**atabase. Using the server tools you can also monitor server processes from another PC.

If you know the task manager from the Windows NT or 2000 environment, you will very quickly familiarize yourself with the server tools.

The ServerPool is part of the server process "EPServerTools.exe"; it is needed for the ServerPooling functionality.

The following table provides an overview of the basic functionality of the server process "EPServerTools.exe":

Table 1: Server Tools Process

Server Tools	Server Pool
<ul style="list-style-type: none">▪ Collects the performance data of DPE server processes and keeps them in a cache.▪ Transfers process performance data to the server tools client.▪ Provides termination services to shutdown single processes and classes of DPE server processes.	<ul style="list-style-type: none">▪ Transfers machine and process load data to the Pooling Server.▪ Starts and assigns IPDServer processes to connecting PPR clients following the server sharing policy.▪ Observes the availability of the IPDServer processes.▪ Terminates unused IPDServer processes to free machine resources.▪ Optionally, logs the Server Activities (e.g. IPDServer start and termination).

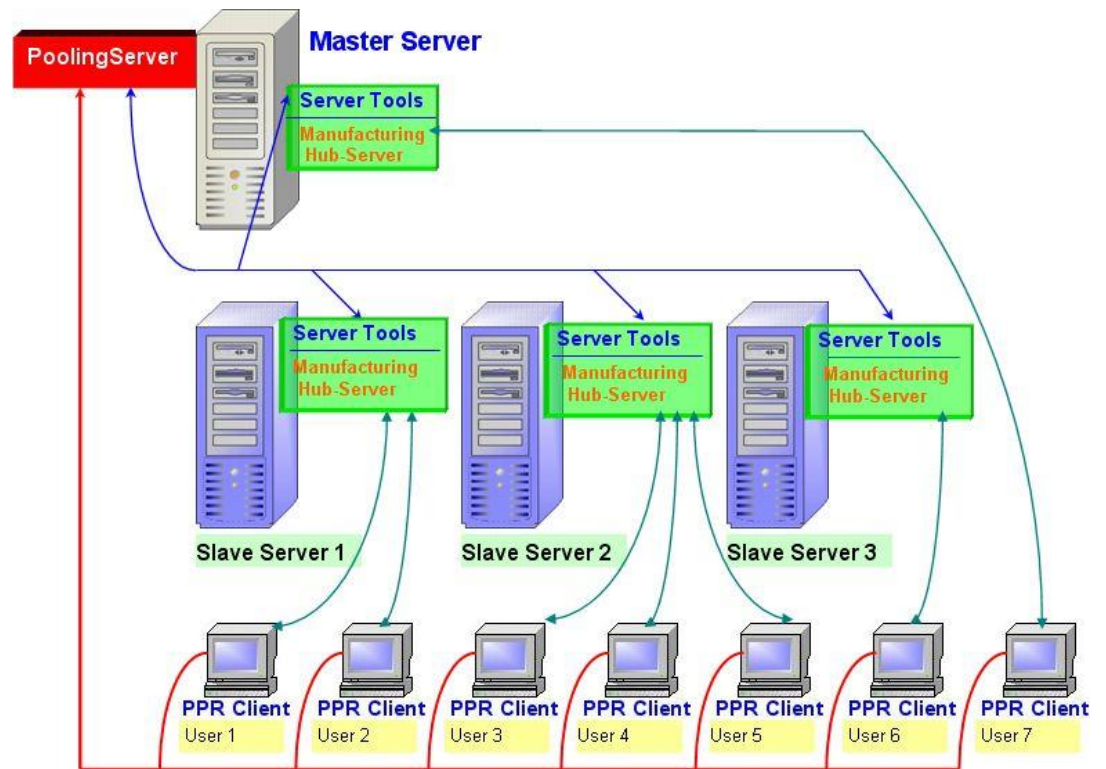


Figure 5: Client-Server Network

4. Logging onto PoolingServer and Server Pool

To view the DPE server installation with all server machines and processes you need to login to the PoolingServer from the ServerTools client. Also, you can login to a specific ServerPool running on a server machine.

4.1 Starting the ServerTools Client

You can start the Server Tools Client as in Delmia Process Engineer either using the Start menu or by double-clicking the appropriate icon on your desktop. To view the Process Engineer icon on your desktop, you may, for example, link it to the desktop using the start menu.



Figure 6: Starting Server Tools using the Start Menu

After starting the program, an empty browser interface appears, the server tools interface.

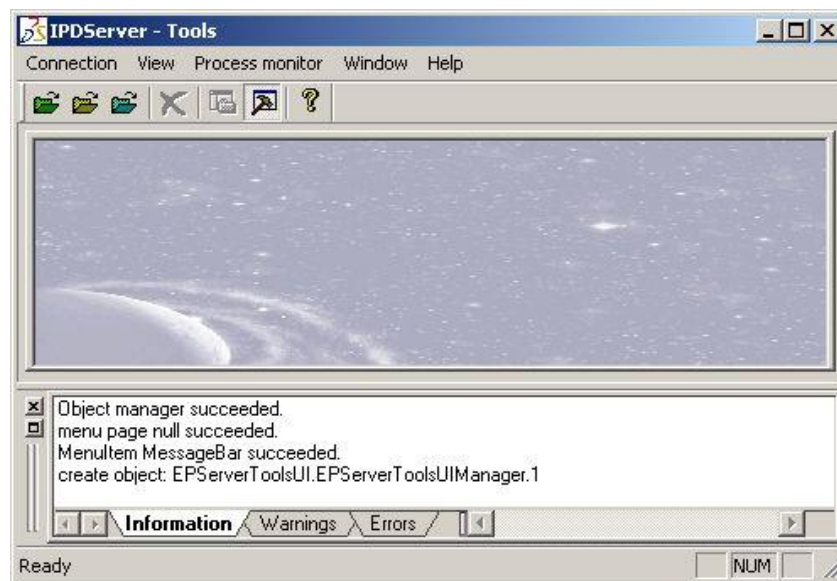



Figure 7: Interface when Starting the Server Tools

In order to logon to the server you must select *Connection* in the menu bar and activate **Connect...** in the menu that opens, or you can activate one of the icons in the tool bar: .

- The **Connect** dialog will open.
- You can connect to the PoolingServer using **Connect PoolingServer** to view the whole server installation with all IPDServer processes.

- Using **Connect ServerPool** you can connect to the Server Pool of any server machine to view a single ServerPool of the server installation containing only the IPDServer processes of that machine..
- Connect to the Lock manager with **Connect LockMng**. A dialog opens in which the locked clients are processed. This function is available only in connection with V5. An object must have been opened in V5.

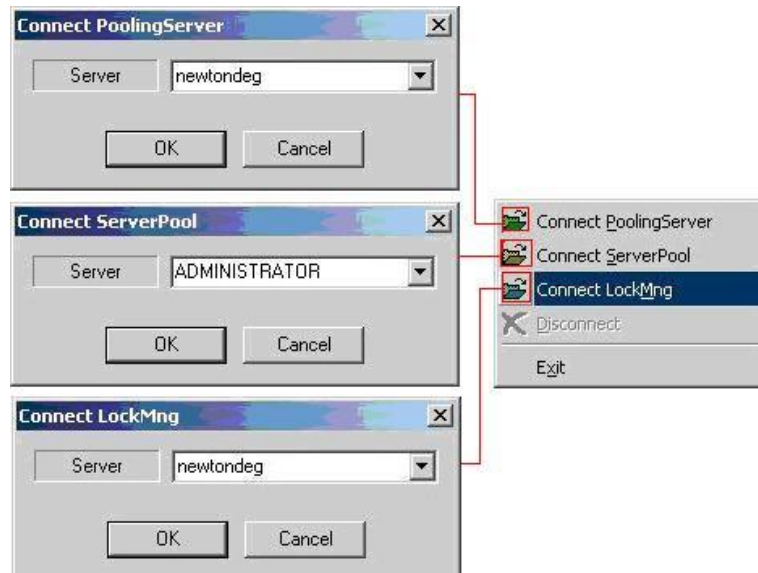


Figure 8: Connect PoolingServer, Connect ServerPool and Connect LockMng dialog box

Entering the Server Name

When logging on for the first time, you have to enter the name of the PC which you want to connect to.

- 1) Then you can select the server name from a list.
- 2) Click **OK** button.

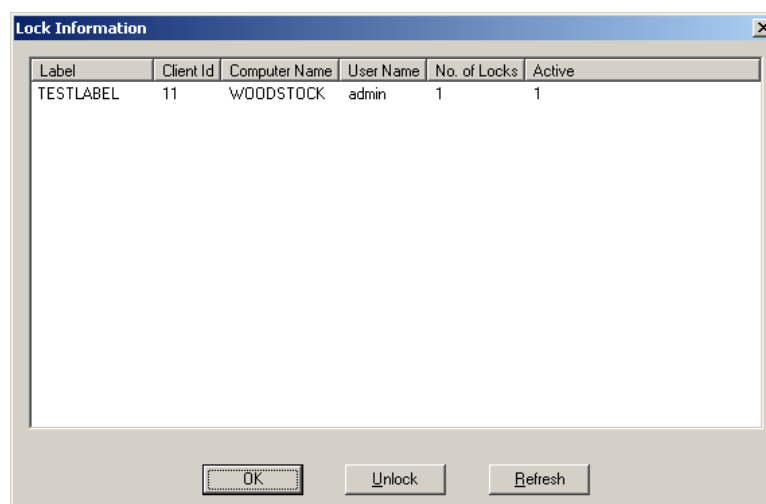


Figure 9: Dialog Component Lock Information in which all Locked Transactions are Performed

- Only locks of inactive transactions can be released. If the client crashes, its transactions will be closed by the DCOM Carbage Collector within 6-8 minutes. Persistent locks can be deleted for the crashed client only after this time period.

- The computer name and user name are only available if there are any persistent locks.
- "No. of Locks" means the number of objects which are locked persistently under a specific lock label.

4.2 Menu Bar and Toolbars



Figure 10: Menu Bar

The Connection Menu

Using this menu you can connect to a server, disconnect the connection or you can leave the server tools.

The View and the Windows Menus

Using these menus you can control the server tools interface.

The Process Monitor Menu

- **Show all Processes** If this entry is check marked, all processes that are running on the PC are shown. You will find the same processes here as in the task manager. If the checkmark is not set, only IPD processes are shown.

Remark: If enabling all processes the view refreshing can be delayed if a large number of processes run on the server machine.

- **Update Speed:** Here you can set the update speed.

The Help Menu

You can use this menu to obtain current Information about the installed version of server tools.

4.3 Working with Server Tools Client

After you have connected to the PoolingServer using **Connect PoolingServer** you can see all of the servers that have been started.

After you have logged on using **Connect ServerPool** you see only the IPD-Server processes of the server machine connected to.

To show clients that are linked to a server, proceed as follows:

- 1) By double-clicking the server or by calling the context menu again and selecting the **Reload** function, all client data is shown.



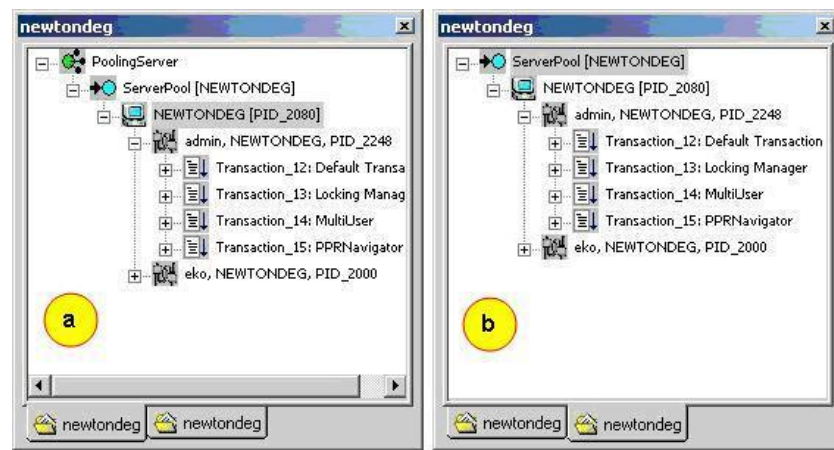


Figure 11: Server Tools with 3 Logged on Clients

Once a node is expanded changes will be automatically updated for the displayed components.

4.3.1 Explore the Server Installation



To display all servers in the ServerTools client, proceed as follows:

- 1) By double-clicking the PoolingServer or by calling the context menu again and selecting the **Reload** function, all Server Pools will be listed under the PoolingServer node.
- 2) By double-clicking the server or by calling the context menu again and selecting the **Reload** function, all IPDServer instances running within these ServerPools are shown.
- 3) Remark: Below the ServerPool node only IPDServer processes are considered which are still available for client connections. If a IPDServer process becomes unavailable or is set to the unused state it will not be displayed anymore as part of the ServerPool. These IPDServers are shutdown candidates and will be terminated the next time depending on their internal closure state.

The navigator can also be expanded by clicking the + symbol on the node.

4.3.1.1 Expansion Hierarchy

The following nodes are displayed by each next expansion level:

PoolingServer

|- ServerPool [<machine>]

|- IPDServer instance [<process id>]

|- connected client <username>, <machine>, <process id>

|- client transaction

Example, displaying all expansion levels :

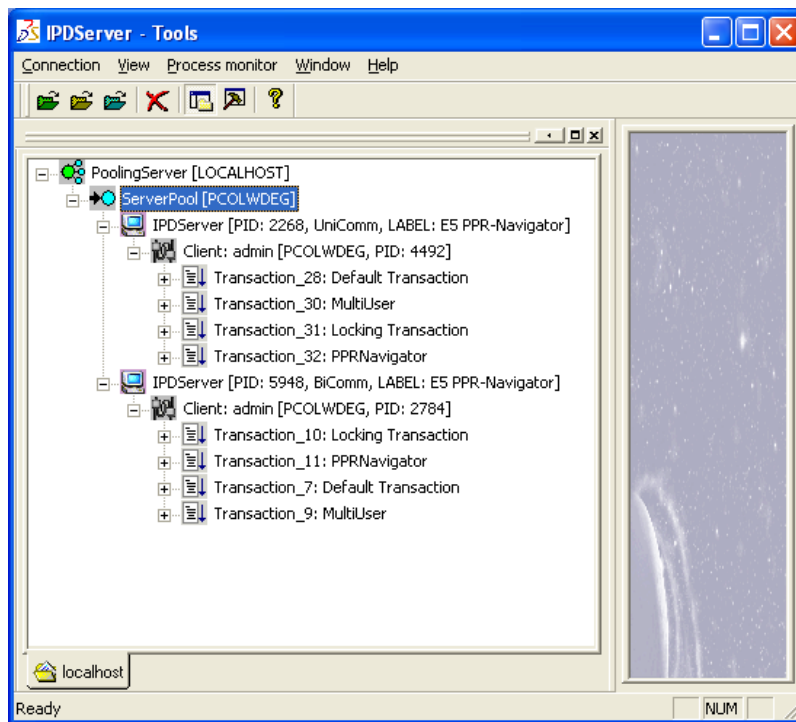


Figure 12: PoolingServer and Server Tools with 2 logged on Clients

4.3.2 Monitoring the Server Installation

With the two main monitor views “Loadindex” and “Process Information” the administrator can have an overview about all E5 server processes running on the server machine and about the current relative load status of these machines. *Please refer to the [Show ServerPool Load Indexes](#) and [Show Process Information](#)”.*

One more important server status to monitor is the emergency error and the lock state of an IPDServer process. Whenever an IPDServer process enters the emergency state caused by a critical error the server process node text in the ServerTools Client will reflect this status change.

There are two main IPDServer process status changes notified in the ServerTools Client: locked and emergency state.

The state “locked” means the IPDServer process is locked for connecting a new client. This lock state can be raised by an emergency error or by administrative functions.

An emergency error is a critical error that can lead to further errors (e.g.: low memory, critical server exception e.g. access violation, critical database exception on Poet errorcodes 6000-6199). If such a critical error occurs the affected IPDServer runs the emergency mode in which the previously connected clients can complete their transactions and new clients can no longer connect to this process.

If an emergency error is caused by a central service used by all server machines, like e.g. the database, the next started IPDServer process could even run into the emergency state if the problem persists. In such case further investigations are needed to analyze the problem cause to allow new clients to continue after connecting.

Example

The following example presents the two states of the IPDServer process 1828 on PYRAMID:

- Connectable IPDServer process 1828 on PYRAMID:



Figure 13: IPDServer process 1828 on PYRAMID:

- IPDServer process 1828 locked for a new PPR client connection on PYRAMID.

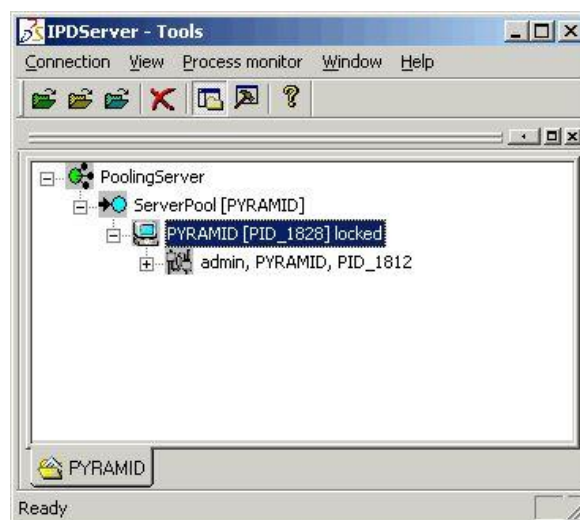


Figure 14: IPDServer Process 1828 Locked

- The locked condition is shown upon refresh (“Reload”) of the machine node or automatically if the node was already expanded before.

4.3.2.1 Resource Alerts

A set of resource indicators continuously monitor system resources to detect potential resource overloads. This new resource control mechanism monitors available physical memory, available virtual pagefile memory, server machine CPU utilization, the number of active IPDServer processes, the total number of processes, and total number of allocated handles.

When a resource alert is raised for a machine:

- It is logged in the PoolingServer.log and the ServerTool.log file
- The lock state is displayed as ServerPool node text

When a resource alert occurs, a client can not connect to the locked Server-Pool as long as the resource overuse continues.

The ServerTools client displays the pool resource lock with the following tree node text:

locked <reason>

where <reason> is a placeholder for one of the following resource lock causes:

- Too many IPDServer processes in pool
- Too many processes on the machine
- Too many handles allocated on the machine
- Insufficient physical machine memory
- Insufficient virtual machine memory
- Machine CPU utilization too high

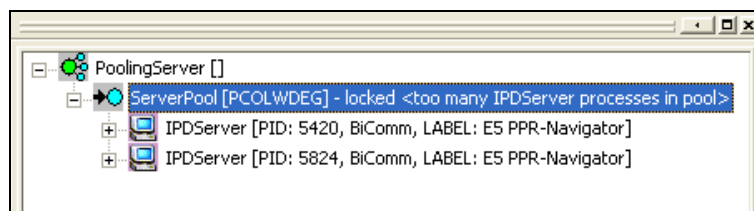


Figure 15: ServerPool Configuration

If all of the ServerPools configured for the connect schedule of a specific client in the PoolingServer configuration are resource locked, the client cannot connect and displays an error message:

- “The server pools are currently locked by resource alert.”
- “There is no ServerPool available to connect the client.”

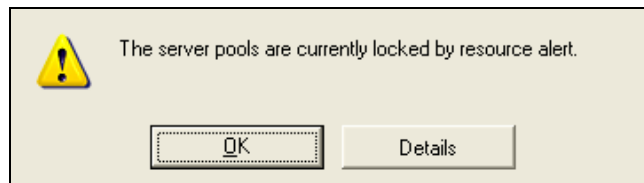


Figure 16: Error Message

When this occurs, either the user has to wait until the machine resource consumption has fallen below the resource threshold value, or an administrator must change the PoolingServer configuration to schedule additional alternative machines. The resource threshold defined should match the actual machine capacity.

4.3.2.2 Viewing Resource Utilization

Current resource utilization can be viewed via the “PoolingServer resource Utilisation Monitor”. To open it, select the context menu “Show ServerPool resource utilisation...” in the ServerTools client on the PoolingServer node:

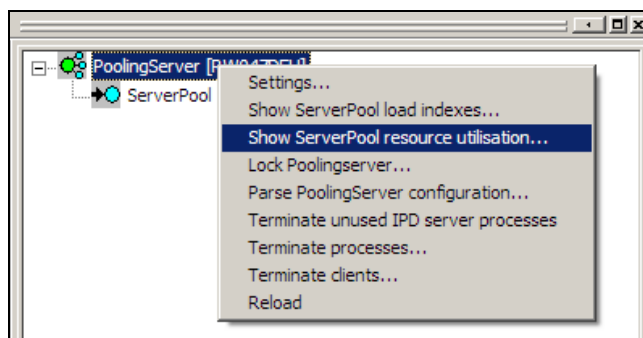


Figure 17: ServerPool Resource Utilization

In this view, the utilization values of machine resources are displayed in percent values compared with the defined limits per ServerPool in each row:

PoolingServer [RW047DEU] Resource Utilisation Monitor					
Pool Id	Machine	Pool avail.	Processes in Pool [%]	Processes total [%]	Machine CPU [%]
1	RW04...	Yes	0	4	7

Figure 18: Resource Utilisation Monitor

- Pool Id
- Machine name
- Pool availability
 - Yes (if the Pool is still available for client connect schedule)
 - No (if the Pool is lock by resource alert)
- Active IPDServer processes in Pool
- The total number of processes on the server machine
- Machine CPU utilization
- Machine physical memory utilization
- Machine virtual memory utilization
- Total handles of the server machine

When any of these resource utilization values reaches 100%, the resource alert for that resource indicator will be raised and the ServerPool will remain locked as long as the resource is utilized equal or greater 100%.

4.3.2.3 Setting Resource Alert Threshold Values

The threshold values of the resource alert metrics can be customized by an administrator via the Windows registry on each server machine, or using the Settings dialog of the ServerPool in the ServerTools client. The relevant registry entries are described below.

Key: [HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\Ergoplan EPSTools\serverpool]

Table 2: Threshold Values

Name	Default	Description
pool_processlimit	100	maximum number of active IPDServer processes in the ServerPool

Name	Default	Description
machine_totalprocesslimit	1000	maximum total number of processes running on the machine
machine_totalhandlelimit	100000000	maximum total number of handles allocated on the machine
machine_availablephysmemory_kb	100	minimum amount of available physical memory on the machine (see Windows Task Manager > System Performance tab)
machine_availablevirtmemory_mb	100	minimum amount of available virtual page-file memory on the machine (see Windows Task Manager > System Performance)
machine_cpu_utilization	100	maximum server machine CPU utilization (in percent) being allowed for the server-pooling, can be specified as decimal value

Key: [HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\Ergoplan EPSTools\serverpool]

- memorylimit_available_mb

This registry entry for the ServerPool memory limit supports memory limitations for processes running under different versions of Windows, and specifies the process memory limit for new client connections. If the amount of available virtual process memory is exceeded, the IPDServer process will not be scheduled for new client connections.

The possible value range for memorylimit_available_mb is 0 to 2147483647. The value selected for this entry is dependent on the administrator's experience with the hardware and the operating system version used to avoid machine overload.

If a single resource indicator is intended to be effectively disabled for the pool resource alert management, its threshold value must be selected with a very high or low value (depending on the kind of indicator) to prevent raising a resource alert. Examples of such disabled values are:

- pool_processlimit = 100000
- machine_totalprocesslimit = 1000000
- machine_totalhandlelimit = 1000000000
- machine_availablephysmemory_kb = 1
- machine_availablevirtmemory_mb = 1
- machine_cpu_utilization = 1000

The memory indicators 'machine_availablephysmemory_kb' and 'machine_availablevirtmemory_mb' cannot be completely disabled. However, these thresholds can be reduced to an extreme minimum so that they can be as ineffective as possible.

4.3.2.4 Turning off Resource Alerts

If the resource alerting capability needs to be switched off, the following registry key should be used:

Key: [HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\Ergoplan EPSTools\serverpool]

- disable_resourcealert

Note that this registry key is not created automatically, and must be manually entered in the registry.

4.3.3 Changes in R18

4.3.3.1 IPDServer Communication Mode

In the IPDServer node text the current connection mode is displayed. There are two cases:

- 1) The clients are connected bidirectional to be updated on data changes using callbacks
 - node text [PID: nnn, BiComm, ...]
- 2) The clients are connected unidirectional without the use of callbacks
 - node text [PID: nnn, UniComm, ...]

4.3.3.2 Interrupted view Refreshing

If the server connection of a monitor view, e.g. Process Information or Load Index, is interrupted the view cannot be refreshed anymore. Since R18 an error message is displayed instead to let the view in frozen state:

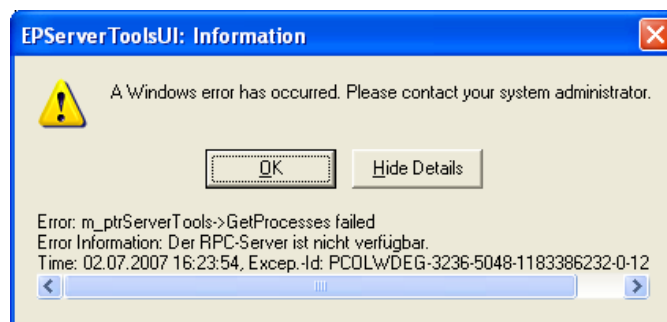


Figure 19: Error Message

The view will contains an error message like in the following example:

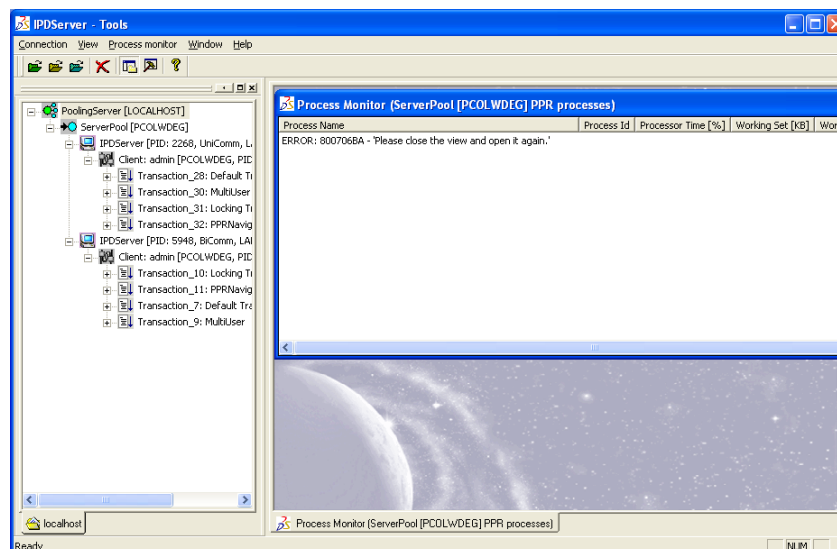


Figure 20: View will Contains an Error Message

4.3.3.3 Parse PoolingServer Configuration

With this function any Poolingserver configuration can now be checked before setting it productive. *Please refer to the [“Parse PoolingServer Configuration”](#).*

4.4 PoolingServer Context Menu

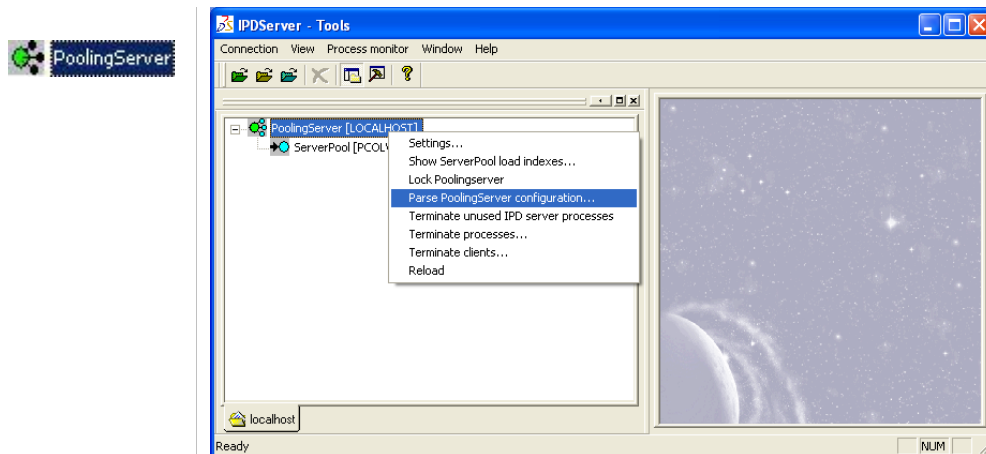


Figure 21: Context Menu of the Pooling Server

4.4.1 Settings

With the context menu “settings...” the runtime parameter of the PoolingServer, like the loadbalancing computation settings, can be edited during operation. Detailed information about how to change these parameter can be found in chapter PoolingServer section settings.

If you confirm the setting changes with **OK** the new values will be used in the PoolingServer.

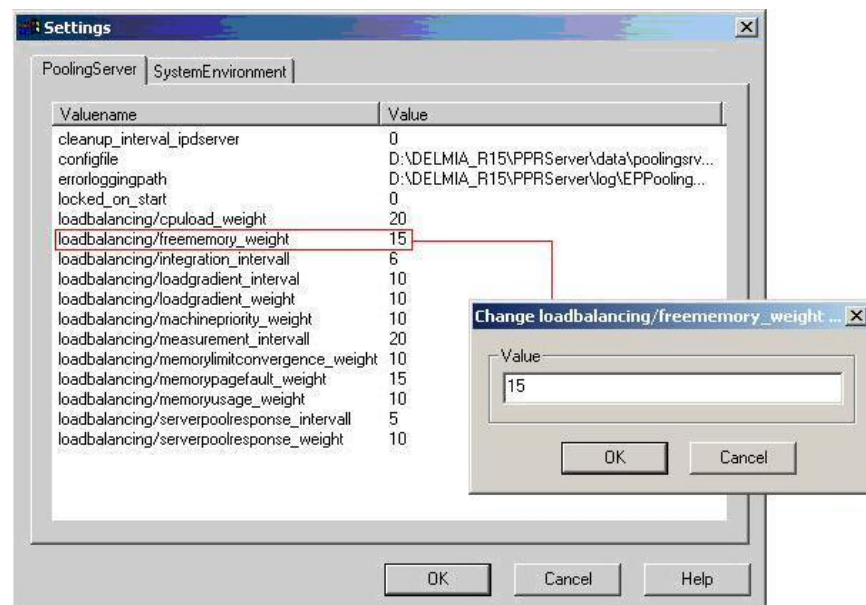


Figure 22: Properties of the PoolingServer

With the second tab of the settings dialog the runtime parameter of the DPE System environment, like security settings, can be changed.

These parameter will be used for the whole DPE installation and require to restart DPE by terminating all server processes if the changes should become effective.

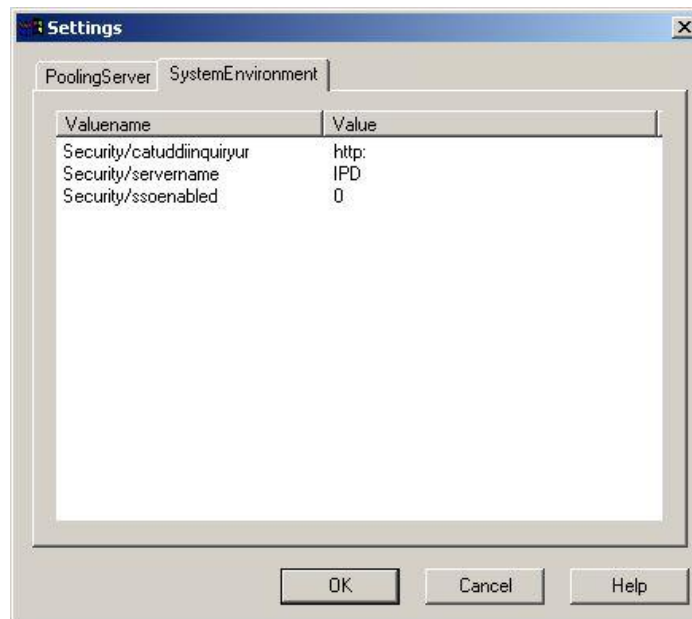


Figure 23: Settings of the Pooling Servers Administration Client

4.4.1.1 Configurable Settings of the PoolingServer

The PoolingServer can be customized by configurable registry settings to the needs of the used installation considering hardware capabilities, amount of user logins, usage of separate programs and tools connecting the PPR server, etc... Using the ServerTools client's Settings User Interface they can be changed even while the PoolingServer is already running.

Detailed information about these configurable settings. *Please refer to the [PoolingServer Settings](#).*

4.4.1.2 Changing Criteria

The standard settings are not critical and are sufficient for usage measurements of servers with normal utilization. Adjustments are only needed if differences in the actual load ratio of the servers relative to each other are not reflected in different load indexes. The cause may be a uniformly disproportionate heavy load on all servers for an individual criterion or a different hardware resulting in disproportional load indicator changes. To correct the load indicator weightings in these cases single performance indicators values must reflect the real load change for that measurement criteria (e.g. CPU utilization)

Since the default load index computation already considers different scenarios a customization should only be necessary for installations with large differences of the hardware equipment like CPU power capacity or harddisk access speed if the machine swaps.

Special PC Configurations

Example

If all servers have too same small free available machine memory (RAM), the PoolingServer decision can be optimized by lowering the **freememory_weight** criteria. The lower weighting means that other criteria become more important for the decision. Consider, this scenario would only be valid if all server machines would continuously swap memory. Due to the heavy impact on the server performance it should be even avoided by installing more RAM.

The same holds for uniformly high memory page faults (**Memorypage_fault_weight**) on all servers. A high memory page fault rate occurs if consistent RAM memory pages are *written to external storage files* when the system has a low memory configuration or if high file IO buffering activity occurs. If this affects all server machines, a lower weighting can improve the Pooling-

Server decision. A large number of memory page faults usually occur only short-term, so that the above-mentioned case is more likely due to a RAM memory configuration that is too low.

In general it can be assumed that the measurement_interval [sec] should be a multiple (≥ 4) of the Serverpool response_interval [sec]. The PoolingServer counts the answers of the server tools to assess the general reliability of the answer time response of the PPRServer in the network.

The effects of changes to the measurement intervals:

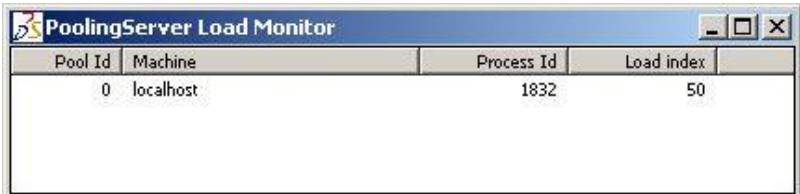
- The higher the Integration_interval [measure-cycles] is set, the less short term measure fluctuations and peaky server usage changes are considered in the balancing decision. The default integration interval guaranties a stable load index calculation reflecting only significant machine load changes. If increasing the integration interval too much the balancing will be imprecise for weakly utilized server machines.
- The higher the Loadgradient_interval [Integ-cycles] that is selected, the more the long-run server machine load trend is considered. If the interval is set too high the balancing will be imprecise for weakly utilized server machines or will react too slow even on high load changes.

The sum of all criteria must equal 100%. The usage index is a relative value with a range of 0 to 100. If machine priorities different to 0 are configured in the PoolingServer configuration that prioritization portion will be added on.

A higher usage index indicates greater machine usage for a PPR Server compared to the other listed server machines.

4.4.2 Show ServerPool Load Indexes

If you select "Show ServerPool load indexes..." a dialog opens in which all of the capacity usage data of the present capacity usage indexes calculated by the ServerPool are displayed.



Pool Id	Machine	Process Id	Load index
0	localhost	1832	50

Figure 24: "Show ServerPool Load Indexes" of the Pooling Server

4.4.3 Lock PoolingServer

Selecting "Lock PoolingServer" prohibits new clients from logging on. Therefore no clients can logon if this is selected. If trying to logon they receive an error message about the connect lock state.

Locking the PoolingServer allows to administer the DPE server installation or the database safely. If the administration activities require to connect special clients, e.g. to import some data, these clients can be enabled to connect using the PoolingServer configuration. Details about declaring administration clients see chapter "The PoolingServer configuration", section "Administration clients".

Administration activities which require to restart the whole DPE server installation including the PoolingServer process can be safely protected against unexpected client connects if the PoolingServer is set in lock mode from the PoolingServer process startup. This behaviour can be controlled by the Win-

dows Registry or the according batch command of the ServerTools Client. Details see chapter "PoolingServer Settings", section "Connect administration".

With **Unlock Poolingserver**, which is in the same context menu, you can permit logging on again.

4.4.4 Parse PoolingServer Configuration

With the in R18 new introduced new menu entry '**Parse PoolingServer configuration...**' of the Poolingserver's context menu, any configuration file can be checked in advanced while another configuration is still productive.

Checking the configuration will be completed by a success or failure message; any error will also be written in the ServerToolsUI.log file.

- 1) In the PoolingServer context menu select Parse PoolingServer configuration.

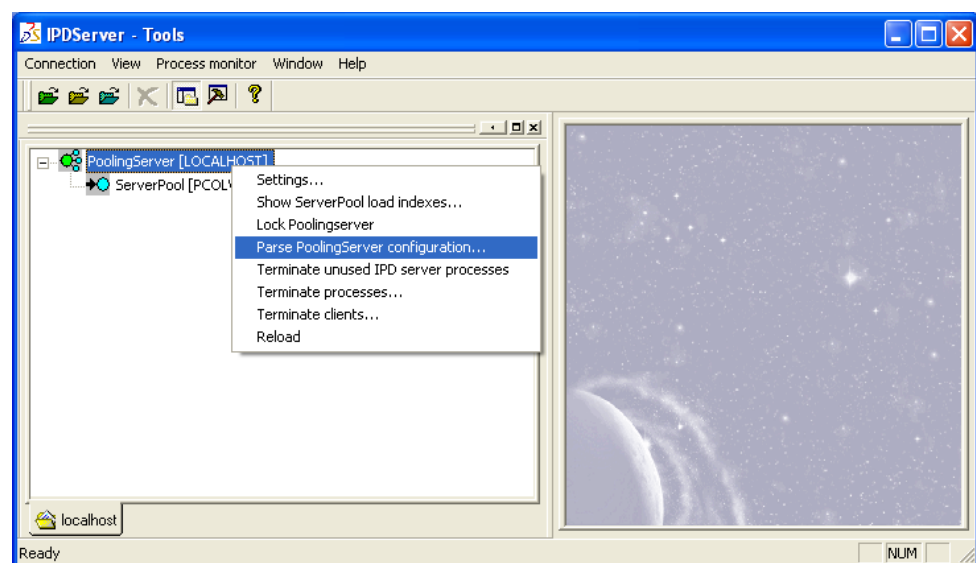


Figure 25: PoolingServer Context Menu

- 2) After opening the Configuration File selection dialog the last checked PoolingServer configuration is pre-selected:

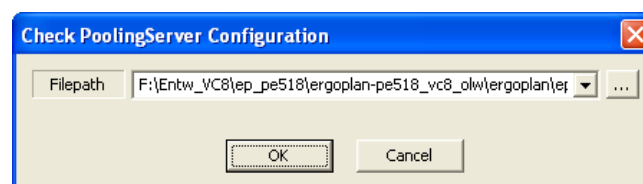


Figure 26: PoolingServer configuration

- 3) Select a previously checked File from the DropDown-Box:

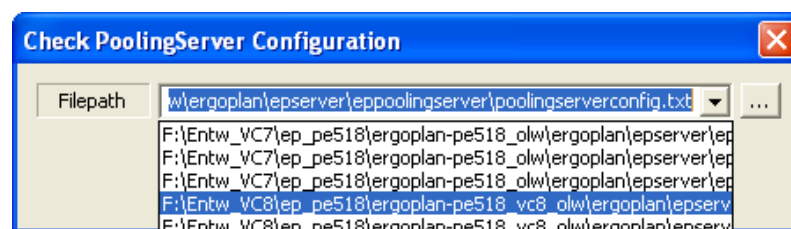


Figure 27: Select File

- 4) If the file to check is not contained in the list of already parsed PoolingServer configurations (item 3) press the button “...” to select the file by exploring the drives:

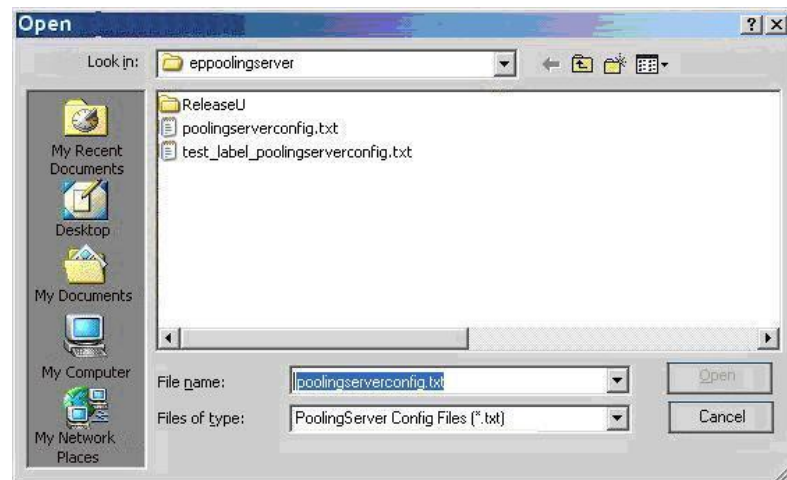


Figure 28: Open Dialog

- 5) If the check has failed an error message is displayed.

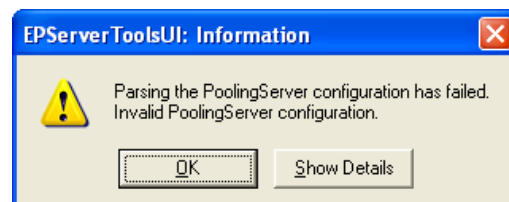


Figure 29: Error Message

Press the button “Show Details” to display detailed information about the cause and the error position in the configfile,



Figure 30: Detailed Information

In this example the token ‘]’ is expected in line 85 of the PoolingServer configuration instead of the present string. If scrolling to the right you can find the last parsed text containing the error position.

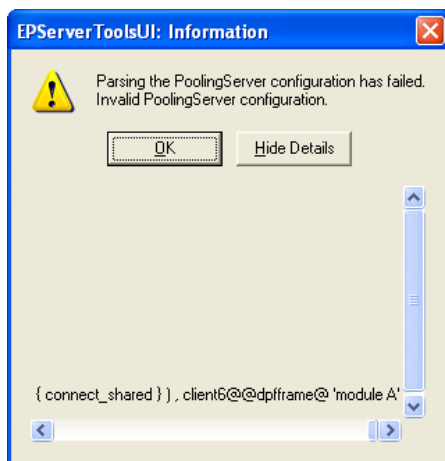


Figure 31: Last Parsed Text containing the Error Position

The complete error description will also be written to the ServerToolsUI client errorlogfile.

- 6) If parsing the PoolingServer configuration has completed with success the number of currently configured server machines is displayed in the succeeded message too.



Figure 32: PoolingServer Configuration Completion

4.4.5 Terminate Unused IPD Server Processes

The function “Terminate unused IPD server processes” is used to terminate IPDServer processes to which no PPR Clients are connected anymore.

The client connection to the server is either released by closing the client regularly or upon terminating the client via the ServerTools Client (context menu *Terminate Client*). Notice, the administrative client termination is only available for PE Clients, not e.g for DPM Clients, *Please refer to the [Terminate Clients](#).*



Note

Disconnecting the PE Client from the from the server, upon call-up of “Terminate Client” in the Server Tool, will only be effective if the message “Connection to server has been terminated” was closed in the PE Client, otherwise the server connection remains in the Server Pool and prevents the IPDServer termination via proceeding “Terminate unused IPD server processes”.

The function “Terminate unused IPD server processes” is proceeded asynchronously to keep the ServerTools Client alive for further user interaction, i.e. if that function is called the work is done in background and takes some time to complete. Also, there are some prerequisites to fulfill till the IPDServer process can be finally shut down. The main prerequisite is to release the DCOM connection of the client in the IPDServer and the ServerPool. If a client crashes, i.e. closes not regularly, it takes minimum 6 minutes till Windows releases the DCOM connection from the crashed client process in the Server

Pool. Depending on other DCOM activities in the affected machines this time can even extend to 20 minutes.

The termination is proceeded in each ServerPool, i.e. for each server machine. The ServerPool first tries to shut down IPDServers which are not connected anymore, whereby every IPDServer process first terminates external DCOM connections and closes its server service routines and then itself.

If this comprehensive shutdown of an IPDServer process does not succeed within 15 seconds, the process is forced to terminate immediately without explicitly closing resources and waiting for service closure.

The function “Terminate unused IPD server processes” can be called up via the context menu or in a batch file.

Batch call-up command:

```
DPFFrame.exe <dpfres>:epserverToolsres.dll</dpfres>
/gc_ipdserver
```

Using the batch command requires to set the following registry entry:

```
key: HKEY_LOCAL_MACHINE\SOFTWARE\Delmia\EPServerToolsUI
entry: PoolingServer
value: <machine>
```

<machine> represents the name of the computer where the PoolingServer is running.

If more than one PoolingServer instance is used, list each of the computer names that are running PoolingServer instances with each separated by comma. For example:

‘ServerPC1, ServerPC2, ServerPC3’

It is recommended to configure the PoolingServer to periodically call “Terminate unused IPD server processes” by the provided automatic cleanup routine in order to release system resources bound to these IPDServer processes as early as possible. This automatism can be configured via the Windows Registry. The setup installation enables it by default with an interval of 5 minutes. More details see chapter “PoolingServer settings”, section “Connect administration” – entry “cleanup_interval_ipdserver”.

4.4.6 Terminate Processes

If you select “Terminate processes”, a dialog appears which enables you to determine whether all PPR server processes or only the IPDServer processes should be terminated.

- Both cases are executed on all server PCs.

When selecting “Terminate all PPR server processes” the Update and Lock manager on the Master Server are terminated too. The EPServerTools and the PoolingServer process are not terminated, since they are needed for the server tools user interface.



Figure 33: Terminate Server Processes

In addition all EPGenericServices.exe, UserExitPlugin.exe, EPSSOService.exe and EPLogger.exe processes are terminated.

If selecting “Terminate only IPDServer process” only the IPDServer processes and its dependent processes like EPGenericServices.exe, UserExitPlugin are terminated.

The following batch commands are derived:

a) `/kill_all_pprserver_processes`

- Terminates all PPRServer processes (IPDServer.exe, LockMng.exe, UpdateMng.exe, UserExitPlugin.exe, EPGenericServices.exe, EPSSOService.exe and EPLogger.exe) on all computers.

b) `/kill_all_ipdserver_processes`

- Terminates all IPDServer processes (IPDServer.exe, EPGenericServices.exe, UserExitPlugin.exe) on all computers.

Example

```
\bin\DPFFrame.exe <dpfres>epservertoolsres.dll</dpfres>
/kill_all_ipdserver_processes
```

4.4.7 Terminate Clients

With this you can terminate all client processes of the DPE installation. *Please refer to the [Terminate Clients](#).*

4.4.8 Reload and Show Running ServerPools

With this function all ServerPool nodes can be reloaded manually if needed.

If the number of ServerPool machines is changed by the PoolingServer configuration or through other administration activities by the PoolingServer itself the view will be automatically updated as far as it is expanded.

4.5 ServerPool Context Menu

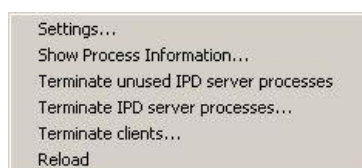


Figure 34: Context Menu of the ServerPool

4.5.1 ServerPool Settings

Using these settings you can edit the registration entries of the ServerTools and the ServerPool running within the EPSTools.exe server process.

The server pool administers the IPDServer processes, whereby certain threshold values, like client and memory limit, are observed for assigning a new server process to the client. Thus, if the limit for number of clients and the utilization of virtual memory are exceeded, a new server will be started. More details see chapter “ServerTools/ServerPool Settings”

The settings that you can enter under the **Server Tools** tab also affect the ServerPool. The server process EPSTools.exe provides the administrative tool functionality as well as the ServerPool instance.

With these settings the EPSTools server process can be customized to the needs of the installation. The changes can be done even if the server installation already runs.

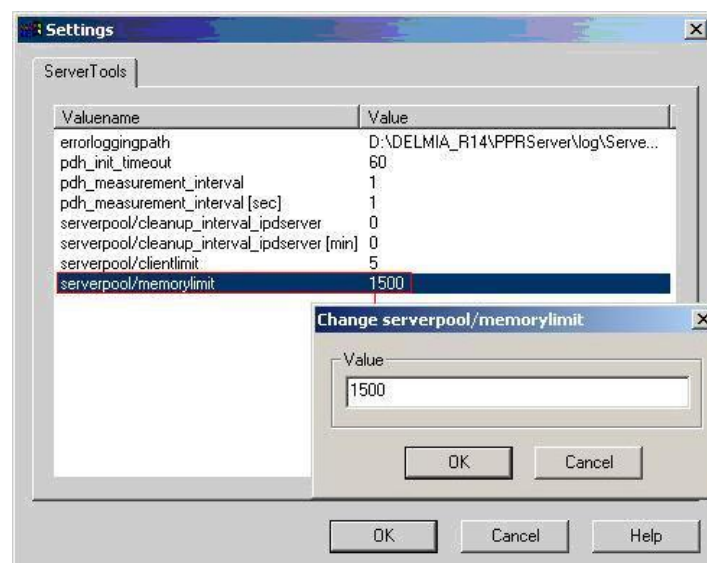


Figure 35: Properties of the ServerPool

If changing settings and closing the dialog by pressing OK they will be immediately effective for the functionality of the EPSTools process - restarting that server process is not required.

If these settings would be changed in the Windows Registry directly the server process would have to be restarted. By that reason it is recommended to change settings always using the ServerTools Client.

Detailed information about the ServerTools/ServerPool settings. *Please refer to the [PoolingServer Settings](#).*

4.5.2 Show Process Information

Calling this menu item opens a view in the right pane of the ServerTools Client displaying server process performance information like known from the Windows Task Manager.

Depending in whether the options “Show all processes” is enabled in the ServerTools menubar a different number of processes of the server machine is displayed in the process view.

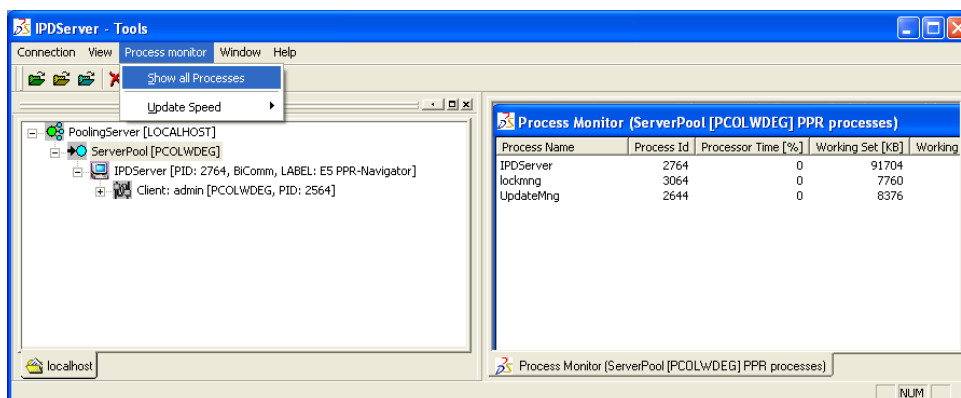


Figure 36: ServerTools Menubar

In the following examples the default mode with disabled “Show all processes” is used. In this case all PPR server processes are displayed – these are on a Slave server machine:

- IPDServer.exe
- EPGenericServices.exe
- EPLoader.exe
- UserExitPlugin.exe
- EPSSOService.exe

Depending on the used server functionality some processes may not appear.

On a Master server additionally the processes

UpdateMng.exe and LockMng.exe are displayed.

The process view displays similar columns like the Windows Task Manager with one exception regarding the process virtual memory. This is the column *virtual bytes (KB)*. It shows the virtual address memory used by a PPR process. The virtual address space of a process is limited to 2GB or 3GB in special Mode for Windows 32 bit systems. A PPR process can no longer operate after this limit is reached. To prevent this, you need to adjust the memory limit, as described in the previous section.

Process Name	Process Id	Processor Time...	Working Set...	Working Set Peak...	Virtual Bytes...	Virtual Bytes Peak...	Thread Count
IPDServer	1832	0	65992	68300	153776	157748	46
lockmng	1676	0	5288	5316	35436	38508	10
rightsadm	1652	0	8800	8888	31932	33980	2
UpdateMng	272	0	6112	6112	32732	32732	7

Figure 37: Show Process Information...

If comparing the Virtual memory column with the Windows Task Manager there is an important difference: the Task Manager displays the virtual private bytes of a process, not the virtual address size. The virtual address size value is always higher due to process memory fragmentation and the system memory management. The essential value for the operability of a process is displayed by the ServerTools client.

4.5.3 Terminate unused IPD Server Processes

Terminate IPDServer processes that are not in use. If this function is called on a ServerPool node it will be proceeded only on the machine where that Ser-

verPool runs. In contrast to, if called on the PoolingServer node the function will be proceeded on all server machines.

Details see the explanation to that function for the PoolingServer node.

4.5.4 Terminate IPD Server Processes

Before terminating all IPDServer processes you should terminate all client processes of this server.



Figure 38: Terminate IPD Server Processes

In addition the server processes "EPGenericServices.exe" and "UserExit-Plugin.exe" are terminated.

4.5.5 Terminate Clients

This is used to terminate all client processes of this server. *Please refer to the [Terminate Clients](#).*

4.5.6 Reload

To manually update the current status and child nodes of the ServerPool execute Reload.

4.6 Server Context Menu

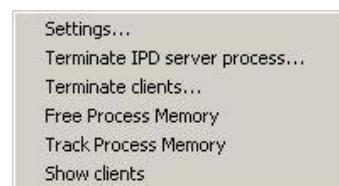


Figure 39: Server Tools Context Menu

Server Settings

The server settings dialog displays the registry settings of the selected IPD-Server as well as the registry settings of the Master processes Update and Lock Manager running on the Master server machine.

Since the settings of the master processes are build by the setup for each server machine they will be displayed even below each Server Pool despite of that machine is the used Master machine. That means, if the settings of the Master processes UpdateMng and LockMng must be changed the settings dialog below the Master ServerPool must be selected to be effective.

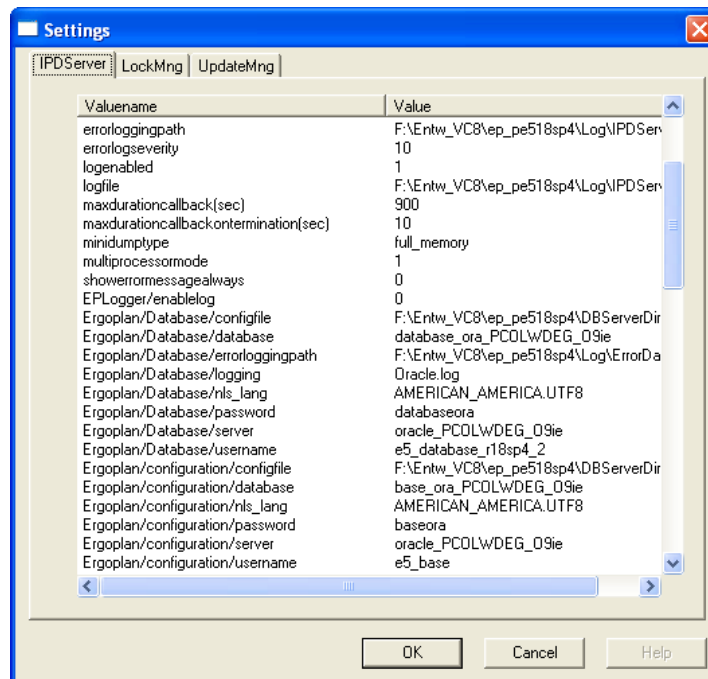


Figure 40: The Server Registration Editor Settings

Terminate IPD Server Process

You can terminate only that IPDServr process being selected.

Terminate Clients

You can terminate all clients that are linked to the IPDServr being selected.

Free Process Memory

Fragmented free memory of the process heap is released with this. The process heap is part of the available process memory. If the Windows Heap manager has already balanced the fragmentation calling that function will have no effect.

Track Process Memory

This is used to display these pages of the working set of the IPDServr process memory that have changed since the last call-up of this function. Two windows are displayed:

- Newly allocated memory
- Memory removed from the working set

The memory required for loading the DLLs is displayed in the window "allocated process memory".

Address	Size [KB]	Owner
007BE000	40	VirtualAlloc/Stack/Other
0204D000	20	D:\DELMIA\program\bin\EngoplanDOImpl.dll
05141000	4996	VirtualAlloc/Stack/Other
05C68000	8696	VirtualAlloc/Stack/Other
090B0000	24	D:\DELMIA\program\bin\GraphicDataps.dll
093BE000	8	VirtualAlloc/Stack/Other
094BE000	8	VirtualAlloc/Stack/Other
095BE000	8	VirtualAlloc/Stack/Other
096BE000	8	VirtualAlloc/Stack/Other
0C24C000	16	VirtualAlloc/Stack/Other
0C44E000	20	VirtualAlloc/Stack/Other
0C450000	828	VirtualAlloc/Stack/Other
0D74E000	8	VirtualAlloc/Stack/Other
0DB4E000	8	VirtualAlloc/Stack/Other
0DD4E000	8	VirtualAlloc/Stack/Other
0DE4E000	8	VirtualAlloc/Stack/Other
0DF4E000	8	VirtualAlloc/Stack/Other
0E04E000	8	VirtualAlloc/Stack/Other
0E14E000	8	VirtualAlloc/Stack/Other
0E24E000	8	VirtualAlloc/Stack/Other
0E34E000	8	VirtualAlloc/Stack/Other
0E44E000	8	VirtualAlloc/Stack/Other
48072000	16	D:\DELMIA\program\bin\p6sm60.dll
4808C000	4	D:\DELMIA\program\bin\p6sm60.dll
4FC13000	12	D:\DELMIA\program\bin\p6sm60.dll
77AA6000	8	C:\WINNT\system32\ole32.dll
77D56000	16	C:\WINNT\system32\RPCRT4.DLL
78027000	4	C:\WINNT\system32\MSVCRT.dll
7FF9C000	4	VirtualAlloc/Stack/Other
7FF8D000	4	VirtualAlloc/Stack/Other
7FF8E000	4	VirtualAlloc/Stack/Other
7FF90000	4	VirtualAlloc/Stack/Other
7FF91000	4	VirtualAlloc/Stack/Other
7FF92000	4	VirtualAlloc/Stack/Other
7FF93000	4	VirtualAlloc/Stack/Other
7FF94000	4	VirtualAlloc/Stack/Other
7FFA0000	4	VirtualAlloc/Stack/Other
7FFA3000	4	VirtualAlloc/Stack/Other
7FFA7000	4	VirtualAlloc/Stack/Other
7FFDD000	64	VirtualAlloc/Stack/Other

Figure 41: Track Process Memory

With this function can be analyzed what memory blocks are allocated at what addresses if a certain client function is called. Only used as a part of administrative in-depth memory analysis.

Reload

To manually update the current status and child nodes of the IPDServer node execute Reload.

4.7 Context Menu on the Client

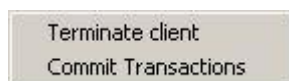


Figure 42: Context Menu on the Client

Terminate Client

You can terminate the selected client process with this.

Commit Transactions

If a client cannot be served any longer (e.g. if it does not get any answer from the server), the data that has not been saved yet by committing open transactions can be saved with this.

5. Terminating Clients

To shutdown PPR server processes without hazarding still connected clients the administrator needs a remote termination method which captures all clients still connected to the server processes to terminate. With the “Terminate Client” and “Terminate server processes” context menus on different hierarchy levels of the server installation the ServerTools client offers such capability to terminate both client and server processes from a central administration console.

Terminating clients is restricted to PE (Process Engineer) clients. Other clients like DPM cannot be closed with this method. DPM even implements another server access architecture which handles a server shutdown by a clientside reconnect. Other clients like import programs for batch processes do not support user interaction. In general, proceeding “Terminate clients” before terminating server processes should be sufficient to close the installation by the cleanest way being applicable.

5.1 Procedure

5.1.1 Starting Clients Termination

The termination of a PE Client can be started from the following nodes:

- **PoolingServer.** All clients of the DELMIA Process Engineer® installation are terminated on the PoolingServer.
- **ServerPool.** All clients of the selected server machine are terminated on the ServerPool.
- **Server machine.** All clients of this Manufacturing Hub server process are terminated.
- On the **client** itself. Only this client is terminated.

Two standpoints must be taken into consideration when clients are shut off.

- The standpoint of the administrator who terminates the clients.
- The standpoint of the user whose client is terminated.

If the administrator selects to terminate a client the following appear:

- A dialog in which two time specifications are possible. The default values are 10 minutes each.

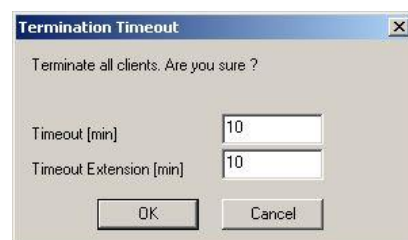


Figure 43: Termination Timeout

- The administrator sets the time span after which the client is to be terminated with the time specification Timeout.
If the value 0 is entered here, the immediate termination of the client is

started, whereby open client transactions are automatically saved. The client user can optionally close the application within 30 seconds and thus decide whether the last changes should be saved or not. If a value of >0 is entered, the client user can select amongst several termination modes for saving transactions (see below).

- The second time specification (Timeout Extension) indicates the time span available to the user to delay the termination. This means that the user can use this time span to delay the client termination. If the default values are used, the user can delay the termination of the client by 20 [min]. If the second target value is set to 0, the client is terminated after 10 minutes, and it is impossible for the client user to influence the termination time span.

The timer extension becomes ineffective if for the Timeout value 0 is specified since the user has no time to use the possible extension. To be applicable by the client user the Timeout must be minimum 1 minute.

The termination of the client is active only after the dialog has been left by clicking OK. The administrator receives a message after the client has been terminated. This message is generated for every ServerPool.

Since the client termination is proceeded asynchronously the ServerTools client can be used further even as long as a client termination runs. If a client termination is running for a certain node in the ServerTools Client the according context menu entry is displayed greyed out and remains inactive till the execution has completed.

- A dialog which refers to the upcoming event appears for users whose clients have been terminated. The users can select various options depending on the time specifications of the administrator.
- A dialog with the termination mode and the time remaining in which the termination can be influenced is displayed to all users whose clients are being terminated.

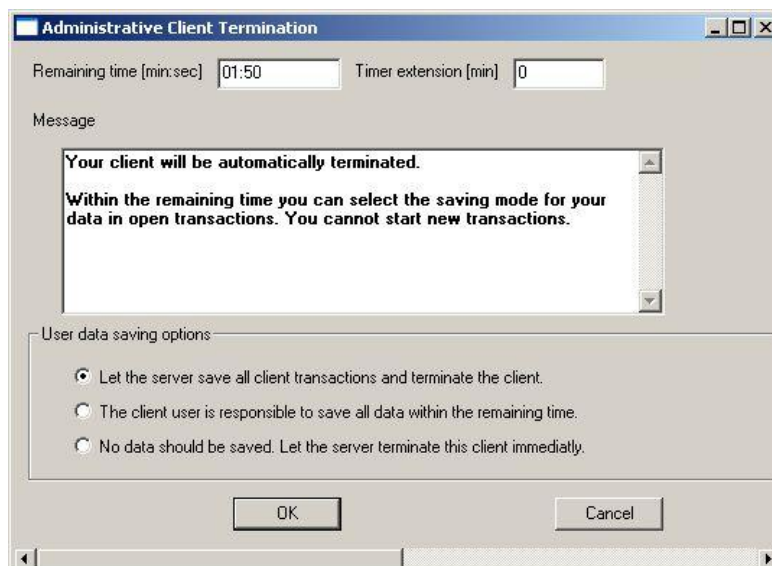


Figure 44: Note on Terminating the Client

The user sees both times specified by the administrator in the upper part of the dialog. The remaining time can not be directly influenced. The remaining time can be indirectly influenced via the *Timer extension*. This is possible only if the administrator has permitted a *Timer extension*. If this is the case, the remaining time is increased by the increment entered in the *Timer extension*



field and activated via the *Apply* button. If the *Apply* button is not visible, the size of the dialog must be increased.

Note

If you do not extend the termination process on time, you will receive a subsequent message.



Figure 45: Termination Process Message

Only the entries that are processed in the remaining time previously displayed by the server are taken into consideration.

You can set the saving options in the lower part of the dialog.

- The server saves all open transactions
- The client user is responsible for saving the data; the server does not save any transactions.
- No data are to be saved. The server does not save any transactions. The user can close the dialog -- this does not stop the termination.

The saving options as well as the times can be switched by pressing the *Apply* button. The dialog does not have to be closed in order to activate the selection.

You can close or leave open the dialog during the termination.

- If the remaining time runs out, the dialog appears for another 30 seconds. If the dialog was still open, only the text in the output window changes – the selected saving option is displayed. The saving options can now no longer be changed; it is, however, still possible to save transactions.

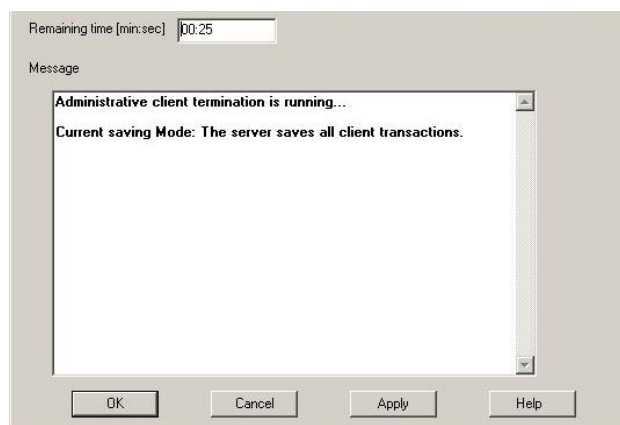


Figure 46: Note on Terminating the Client

- After the 30 seconds have elapsed, a message window appears which indicates that the connection to the server has been terminated. There are no longer any possibilities for saving; the dialog can be ended only by clicking on OK or Cancel. The application is closed in both cases. If the user does not close this dialog it is closed automatically after 30 seconds to ensure disconnecting the client from the server completely even if the user is not on his workplace to interact.

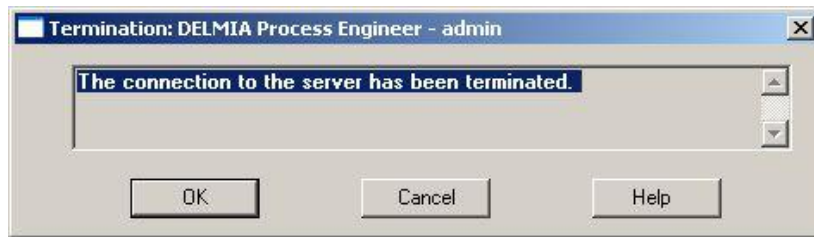


Figure 47: Message (to the client user) that the Client has been Terminated

- The administrator now receives the message about the termination of the client. The message appears for every ServerPool separately if the client termination was started via the pooling server node.



Figure 48: Server Tools Message (to the administrator) after a Client has been-Terminated

6. EPUnlockTool

6.1 General Information

Up until now, permanent data restrictions (locked files, will be referred to as persistent *locks* in the following) could be lifted only with the help of the server tools. The Server Tools are administration tools and they can not be accessed by every user. Persistent locks for a certain user must be deleted if these data are not to be edited any further and the writing rights to them have to be granted to other users. This applies also to the user who triggered the data restriction if the application crashes or the server connection fails and the client has to be reconnected to the server. Persistent locks are only effective as long as the Master server runs, i.e. if the DPE server installation - including the Master server processes - is shut down, these locks are no longer effective.

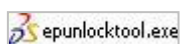
The PPRClient tool "**EPUnlockTool**" gives you the opportunity to release inactive persistent locks e.g. after a client crash. With it previously restricted data can again be edited. This effects especially V5 users since only the system administrator could release locked files after a crash. The new client tool make it possible for the user to release one's own persistent locks and thus to make this data accessible to oneself and other users.

The EPUnlockTool is a client program without its own user interface. Error messages and other information are issued in message dialogs. Errors are furthermore written to the log file "EPUnlockTool.log" in the client log folder.

6.1.1 Mode of Operation

- The client user must always login with a user name and password. The login has the usual PPR client login functionality.
- The user confirms the deletion of the assigned persistent locks in a dialog. Then inactive persistent locks assigned to the user and used computer are deleted.
- Active persistent locks connected to open transactions can **not** be deleted. In order to ensure the data integrity within the DPE, it is important for the client not to be able to delete these active locks.
- If the client crashes, its transactions will be closed by the DCOM Carbage Collector within 6 minutes. Persistent locks can be deleted for the crashed client only after this time period with the "EPUnlockTool" when they become inactive.
- If the deletion is not successful, an error message is displayed.
- All errors are saved in the logfile 'EPUnlockTool.log' and are located in the PPR Client Log folder.

6.2 Operating Manual



- 1) Open the folder of your DELMIA Process Engineer® Installation. Under `..\DELMIA\PPRClient\program\bin` you will find the executable file **epunlocktool.exe**.
- 2) Start the file **epunlocktool.exe** in order to lift restrictions.

- The DELMIA Process Engineer® login dialog opens.



Figure 49: Login Dialog

- 3) After the login the only interactive dialog appears.



Figure 50: Interactive Dialog

- If the button **OK** is pressed, all active permanent locks for the logged in user and the computer are deleted.
- If the button "**Cancel**" is pressed, no permanent locks are deleted.
- All inactive permanent restrictions are released.
- If not all inactive permanent restrictions are released, the number of remaining active locks for the user on the present computer is displayed in an information dialog.

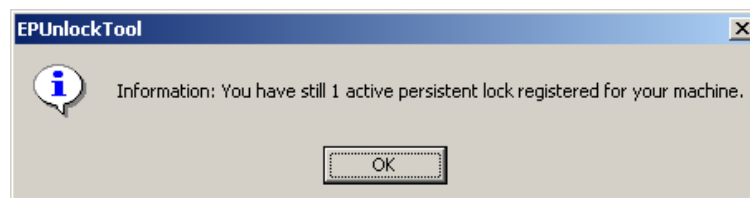


Figure 51: Information Dialog

6.2.1 Release Persistent Locks through Slave Server

The EPUntlocktool provides the capability to communicated with the Master server even if the Master server runs in a different Windows domain and can only be accessed through a firewall from another machine. The following scenario describes such an environment:

A firewall exists between different Windows domains in one the Master server and in the other Slave servers and the clients runs. The communication across the firewall is very limited and allows only the Slave server machines to connect the Master server machine. The clients are in the same domain like their assigned Slave machine. Since the access on the Master server through the firewall is blocked for all clients, a client cannot access the Master directly.

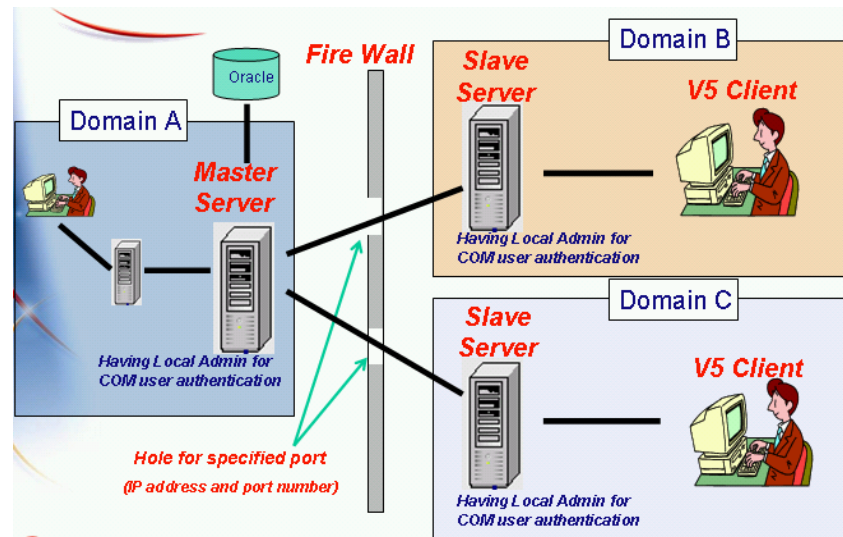


Figure 52: Release Persistent Locks

To overcome that restriction the EPUnlockTool delegates the access to the Master server Lock Manager through an IPDServer process connected temporarily for this purpose.

The firewall must be configured to allow the DCOM communication between Master and Slave server explicitly. The communication between Master server and client machine must be disabled.

Limitation

The functionality of the EPUnlockTool will only be available in an environment protected with a firewall if the firewall allows the DCOM communication between Slave and Master server. If using a separate database machine, the communication between all E5 server machines and the database machine must be allowed additionally.

6.3 Support for Command Line Interface

EPUnlockTool provides support for Command line interface (CLI).

Possible CLI options

/UnlockPersistent

Default behavior

`<USERNAME>xxxxx</USERNAME> <E5_PWD>zzzzz</E5_PWD>`

For silent log-in of EPUnlockTool. User name and password should be in above format.

Other command line arguments should NOT be inserted inside the format. With this enhancement it will be possible for the user to invoke EPUnlockTool through batch.

/Help

Displays help regarding command line arguments. This option should not be combined with any other option.

Example

`EPUnlocktool.exe /UnlockPersistent`

Pops up login dialog. After validating user credentials, unlocks inactive persistent locks locked by user from that particular machine.


```
EPUnlocktool.exe <USERNAME>admin</USERNAME>  
<E5_PWD>admin</E5_PWD>
```

```
EPUnlocktool.exe <USERNAME>admin</USERNAME>  
<E5_PWD>admin</E5_PWD> /UnlockPersistent
```


Does not pop's up login dialog. User credentials are read from command line arguments. After validating user credentials, unlocks inactive persistent locks locked by user from that particular machine.

```
EPUnlocktool.exe /Help
```

Displays help regarding various command line arguments. User credentials not required. This option should not be combined with any other option.

Appendix

6.4 Glossary

S.No.	Term	Definition
1	Administration Clients	Administration Clients are client modules being used for data or infrastructure administration purposes within the DPE installation. These clients need special accesses to the E5 database and server processes. Clients for database administration tasks like data imports have special requirements on system resources (memory, CPU) which should be reserved for these tasks in order to be able to complete them. E5 administration clients: pprloader.exe, dbanalyser.exe, ptimex.exe.
2	Client Net	The assignment of client groups to the server domains.
3	Handles	A value that uniquely identifies a resource, such as a file or a registration key, so that a program can access this resource.
4	Heap	A part of Windows memory management for assigning small memory blocks..
5	IPD Process	IPD processes are the processes that run on the Integrated Process Database of the DELMIA Process Engineer.
6	Memory Page Fault	<p>A memory page fault is triggered when an application tries to access a memory page of the working-set in the main memory and it is no longer entered. In most cases after a memory page fault has occurred, the external storage file or a memory page cache is accessed.</p> <p>The page fault specifies the absolute number of read operations in the task manager. The page fault value increases from the point in time the process is called up.</p> <p>The PoolingServer considers the memory page fault rate per second.</p>
7	Memory Usage 	This is the current workingset of a process in kilobytes. The current workingset is the size of all pages currently loaded into memory for a process. You can find these pages in the task manager under the Processes tab under the column heading <i>Memory usage</i> .
8	Pooling Server	Process on the master computer that measures the performance of the server computers.
9	PoolingServer Configuration File	The Poolingserver configuration file can be used to control the PoolingServer or structure your Client-Server network. The registration key of the configuration file is:...Delmia\Ergoplan PoolingServer\ConfigFile value: <Filepath>
10	Processes: = the operating system process	If 'processes' are often mentioned in this manual, then it is not the processes of the DELMIA Process Engineer Process View that are meant, but the operating system processes
11	Process ID	This is the unique number by which the process is managed.

S.No.	Term	Definition
		A numeric indicator, used to uniquely identify a process during its execution. You can display PIDs in the task manager. PID stands for <i>Process Identifier</i> .
12	RAM or Main Storage	Memory, to which a computer writes and/or reads out information. Information stored in RAM is lost when the computer is turned off. RAM stands for Random Access Memory (main storage). If the main storage is not sufficient for executing a program, additional memory ("virtual memory") must be used.
13	Serverpool	Measures the performance of the individual processes on (its) computer.
14	Server Tools	The Server Tools measure the performance of the individual processes of a client
15	Subnet	Intended sub-area of an existing Client-Server network..
16	Servernet	A Servernet consists of all server domains..
17	Server Domains	For the PoolingServer, a server domain is a group of server PCs.
18	Thread	A program execution path within a process. Threads allow simultaneous procedures within a process and enable a process to execute various parts of its program on different processors at the same time.. The number of threads is displayed.
19	Usage Index	This is a measured value of the PPR Server load. The higher the usage index, the greater the load
20	Usage Data	This includes all performance data involved in the PPR Server usage indexes. The data is gathered on the machine itself on one hand, and from a selected IPD server process on the other. .
21	Usage Distribution	Refers to the most uniform utilization possible of available PPR Server instances. The usage distribution of the PoolingServer refers only to the assignment of new clients.
22	Virtual Memory or Virtual byTes	Temporary memory used by a computer for executing programs that require more memory than is actually available. Programs can access up to 2 GB of virtual memory on the hard drive of the computer, for example, even if the computer only has 128 MB of RAM. Program data that does not currently fit in the RAM memory of the computer is stored in <i>external storage files</i>
23	Working Set Peak	The maximum value of virtual memory. It is the address area of the memory currently used by the process.

6.5 Registry entry

Additional Registry Entry: EPLOGGER

The EPlogger is a server process that will not be directly accessed by the client. The following will be chosen:



Figure 53: Registry Entry Architecture

The intention is to have one EPLLogger Process running central somewhere, but also it is possible to have one on each server machine. To which machine a ipdserver should connect can be setup via registry (see customization section down). The IPDServer will connect to the EPLLogger Process using COM.

The EPLLogger Process will generate XML files in order to store the Log entries.

The new EPLLogger will be a COM process that will be created by the IPDServer when data has to be logged.

The Logging mechanism starts in the server. Logging will be handled in the IPDServer asynchronous. This means the logging request will hand over to the EPLLogger by a separate thread. The caller thread or process will meanwhile continue its job since waiting till the logging has been done. An asynchronous mechanism is required in order to meet the customers performance targets.

If a failure in the EPLLogger will happen Error message on logging request from the IPDserver will be returned. The IPDserver itself will react on such an error by switching to the Emergency mode. The client will then only be allowed to commit or rollback his changes.

The name for the xml files would be in the format:

- **PnO_YYYY-MM-DD.xml** (for P&O Changes) and
- **Export_YYYY-MM-DD.xml** (for Export logging).

The new Xml log files will be created after it reaches the Maximum size limit. If the key is not defined then 30Kb is the threshold for the generation of new file.

The Log File path is stored in the registry

HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\EPLLogger\LogPath.

On IPDServer Side:

The Logging Functionality will be by default deactivated, it can be activated using Registry key

HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\IPDServer\EPLLogger\EnableLog:

- 0 (deactivated, default)
- 1 (P&O Changes are logged only)
- 2 (Only Export Logging Enabled)
- 3 (Everything logged)

The machine on which ipdserver will have to connect in order to connect to the server will be set using the Registry key

HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\IPDServer\EPLLogger\server.

On EPLLogger Side:

- The Path of the Logs is set by the following registry key
HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\EPLLogger\LogPath.

- The Maximum Log File Size can be changed by changing the registry setting stored in
HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\EPLLogger\MaxLogfileSize.
This key is the criteria for generating new log File per day and refers the size in KB, default value is "30" Kb.
- The Path of the error logging is set by the following registry key
HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\EPLLogger\ErrorLoggingPath. Notice that not only the path but the File Name should also be provided e.g. C:\log\eperror.log.
- The error logging is enabled by the following registry key
HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\EPLLogger\ErrorLoggingEnabled.
Any changes done in the registry will not take effect until restart of the IPDServer and the EPLLogger.

Limitation

- Any change done to the logs produced manually by customer may lead to the loss of data.

Data that has to be written is send asynchronously to the EPLLogger. Hence failures in the EPLLogger will be reported to the IPDserver and then to client after a span of time. Loss of logging information can be the consequence.

Summary

HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\EPLOGGER

Table 3: Registry Value Explanations

Registry Value	Value	Explanation
ErrorLoggingEnabled	'1'	- additional error debug logging switched on (1) or off (0), even if set to '0' severe errors will be logged
ErrorLoggingPath	'<install dir> \Log\eplogger.log'	- file path of the error log file
failsafememory	'1024'	- size in KB of the reserved failsafe memory being freed at a low memory error condition
iobuffersize	'256'	- size in KB of the iostreambuffer buffering the harddisc file output
LogPath	'<install dir> \Log\EPLLogger\xml'	- directory used to store the logger output xml files
MaxLogFileSize	'10000'	- file size threshold in KB of a logger output file; the maximum size is considered with the current logger struct size resolution
memorylimit	'800'	- memory limit in MB for new IPDServer connections; if the EPLLogger process working set exceeds this limit new ipdserver will connect to another EPLLogger process

Optional Registry Entry

The Server installation won't create this Registry entry. For the LockMng.exe and UpdateMng.exe the default value after the Setup should be 'normal'.

When critical issues are occurring and advanced Dump-Analysis is required the value 'full_memory' has to be set for the MiniDumpType variable of these processes.

updatemng.exe

key: HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\Ergoplan UpdateMng
entry: MiniDumpType
default value: 'normal'

lockmng.exe

key: HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\Ergoplan LockMng
entry: MiniDumpType
default value: 'normal'

Supported values and mini dump types:

Table 4: Supported Values

Registry Value	Explanation
'none'	no minidump is written
'normal'	minidump with thread stack data
'with_data_segments'	global data sections from loaded modules
'full_memory'	minidump with all accessible process memory
'with_handle_data'	additional information to OS handles
'filter_memory'	with complete memory, but removing ptr values are not necessary not reconstruct the stack
'scan_memory'	full memory, ptr references are scanned

cfcdatatransfer

key: HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\ergoplan
entry: cfcdatatransfer
value: '1'

The default value is '1'.

With this registry entry it will be possible to optimize the clientcache behavior depending on the client functionality currently executed. With the appropriate parameterization of the caching policy, the ConfigFactoryCache can be optimized for server WAN connections by minimizing the number of server calls needed to upload ConfigFactoryCache cache data on client side.

The initial policy value depends on the registry setting:

- Client and Server are running in a LAN network.
The memory requirements for the Client process far less than 2GB. ► value '1'
- Client and Server are running in a LAN network.
The memory requirements for the Client process near 2GB. ► value '0'
- Client and Server are running in a WAN network. ► value '1'
- Client and Server are running in a WAN network.
The client displays property pages with layout information edited by the ConfigTool ► value '7'

6.6 History: Improvements in Server Pooling

DPE5.12

ServerTools-Client

- **Change of administrative termination of E5-Clients (DPFFrame clients only)**
 - Configurable termination timeout
 - Automatic transaction store service (commit)
 - Termination runs in a own thread
- **New menu entry on PoolingServer node: Lock PoolingServer**
 - New client connects will not possible for the DPE installation
 - Allows to proceed administration tasks (outside DPE/DPM)

DPE5.13

ServerTools-Client

- **New menu: terminate unused IPDServer processes (in Service Pack)**
 - Terminates IPDServer which does not have any client connections any more
 - Batch command: /gc_ipdserver

PoolingServer

- **Disable load measurement by Windows performance data (PDH lib functions) for a server machine domain in the configuration**

DPE5.14

ServerTools-Client

- If an IPDServer process cannot be connected anymore (e.g. because running in emergency mode) it will be displayed at the machine node using the text 'locked' additional connect menu 'connect lock manager' providing a dialog with facilities to release inactive persistent locks
Caution: there's an external 'unlock' tool (EPUnLockTool) for the same purpose available.
- **New batch commands**
 - /poolingserver_lock_on_start
 - /poolingserver_unlock_on_start
 - /poolingserver_lock
 - /poolingserver_unlock

PoolingServer

- **Ipdservr garbage collector terminating automatically unused ipdservr processes**
 - The routine is started periodically after a configurable time interval

- **New** configuration feature: client machine not being listed in the PoolingServer configuration can be assigned to the built-in server domain '@anonymous_userdomain'
- New configuration feature: client machines can be assigned to the built-in **client** domain '@admin_domain' to allow connect while the PoolingServer is locked
- New **Registry** entry 'locked_on_start' allows locking the PoolingServer even if the PoolingServer process is restarted.
- With this option administration operations can be proceeded safely with a connect locked DPE installation even if the installation has to be restarted or to be newly started before.

DPE5.15

ServerTools-Client

- **New batch commands**
- /kill_all_pprserver_processes
- /kill_all_ipdserver_processes
- **The process EPGenericServices is killed too if proceeding 'terminate all pprserver processes'**

ServerTools

- **New registry entry 'pdh_init_timeout'**
- the initialization of Windows PDH lib access is proceeded asynchronously using a configurable timeout
- it prevents blocking a client connect while starting the serverpool if there is a PDH access problem

ServerTools/PoolingServer

- Registry entries are changed to names without using [] - brackets
- PoolingServer and ServerPool are initialized with the SystemEnvironment data stored in registry key:
HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\Ergoplan System;
This initialization is mandatory for using the PoolingServer and ServerPool services from other modules

DPE5.16

ServerTools-Client

- **The process EPLogger is killed too if proceeding 'terminate all pprserver processes'**

PoolingServer

- **Increased reinitialisation stability if COM connections to the EPServerTools process are broken; this could be reached by improving the control of the RPC state of remote COM connections to the process EPServerTools:**
- Since the PoolingServer accesses the EPServerTools process by different threads and COM interfaces the management of the RPC errorstate could

be improved by controlling it using a central single representation for each inter-process communication; this optimized handling allows to signal each PoolingServer task the RPC errorstate at the first time it occurs and allows to accelerate the reinitialization and client call throughput; at the same time the risk of blocking thread service routines by pending RPC calls because of broken connections is minimized the reinitialization of a broken RPC connection is accelerated by signaling an PoolingServer reinitialization event immediately after the first RPC error occurs for a remote COM connection

- The improved stability has been checked by a multithreading test simulating 1000 parallel client connects: in this case in R15 killing the EPServerTools process could lead to blocking the waiting client connects if the PoolingServer reinitialization task has been proceeded; with the improvements in R16 this test has been passed
- **New registry entry 'load_distance_threshold allows using a connect count based load balancing strategy if the load difference between pool machines is small**
- A connect timeout prevents blocking other client connects if a single connect hangs while initializing a new ipdserver process in the ServerPool
- This connect blocking over a longer time span occurs also if the initialization of a new created ipdserver process take too much time caused by a lack of resources under high load conditions because the PoolingServer should manage machine loads it must not be blocked by a high load condition in one of its pools
- **Asynchronous initialization of the ServerPooling service: read configuration and connect ServerPools**
- **Automatic retrial of connecting the client:**
 - if There is no serverpool configured for the running client connect, the connect is retried once since the server machine could be reconnected to the same time
 - If a client connect has failed caused by an COM instantiation error while creating an ipdserver the client connect is retried once since a COM timeout could have been prevented the connect while starting up the ipdserver process
 - If an server pool RPC error occurs while connect a client to this server pool, the connect is retried 2 times since the server pool could have been terminated by administration tools causing the RPC error

Server Tools

- **Asynchronous check of the ipdserver availability to prevent blocking the client connect routine if an ipdserver does not respond; with this asynchronous check the multithreading behaviour of the ServerPool has been optimized and allows faster response times if clients are connecting and/or pool information is queried by the ServerTools-Client**
- The availability check uses a timeout; if exceeded, the ServerPool assumes the IPDServer process is not available any more; configurable in registry :

HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\Ergoplan EPServer-Tools\serverpool\ipdserver_availability_check_timeout = 30 // second

- If an IPDServer is set unavailable by this check, it will be terminated regardless if its internal list of running transactions is still not empty!

DPE 5.16SP6 PoolingServer

- **Asynchronous** initialization of the ServerTools process; this initialization includes the creation of the ServerPool and the startup of the *load measurement* via the Win PDH lib
- Where the PDH initialization was delayed (5 minutes) the synchronous server tools initialization became a risk for the client connect to hang up resp. or delay a longer time if the PDH initialization fails by a timeout or hangs the user needs to be informed by an error message
- The client connect timeout is used for the server tools initialization too: HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\ERGOPlan PoolingServer \ client_connect_timeout
- **The** termination of unused IPDServer processes is now asynchronously executed:
- Since a hanging termination call within the ServerPool could lead to a blocked client login, the ServerPool call is proceeded asynchronously
- The use of write events for the ipdserver brokers, forcing the client login to **wait**, has been completely removed for the following common, mostly used pool management use cases:
- The PoolingServer configuration file has not changed
- The ServerPools does not need to be reconnected
the **ServerPools** only needs to be reconnected if the DCOM connection or the load measurements callbacks do not respond this improvement reduces efficiently a large potential for waiting or hanging client logins; due to this enhancement the Broker cleanup time is reduced from 30 sec. to 20 sec. and the PoolingServer configuration change check time from 30 sec. to 15 sec.

ServerTools

- **Create** the ipdserver without initialization (incl. database connect), with this approach the ServerPool can manage even ipdserver instances which initialization will fail later and consider these instances for the ipdserver garbage collector rightly. Before this change the ServerPool has killed all ipdserver instances being not controlled by the current ServerPool instance with the disadvantage that even all used ipdserver instances of a previous ServerPool has been terminated. That could happen if the ServerPooling installation has only be shutdown partially by killing a ServerPool without killing the PoolingServer. This partial shutdown is allowed and even used by some customers an emergency case. If killing a EPServerTools process clients can still continue working with the connected IPDServer instances, even if the ipdserver garbage collector has been enabled.
- **Regarding** the item above, the initialization of the ipdserver (incl. database connect) will be proceeded by a separate initialization method "IEIPDServerInstance::InitailizeIPDServer". The ServerPool calls this

methods asynchronously in order to prevent blocking the ServerPooling if that IPDServer initialization will be hang up, e.g. by a pending, incomplete database connect (e.g. some system resources could be limited and force opening a new database connect to wait). The timeout for the ipdserver initialization now can be configured by registry:

HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\Ergoplan EPServerTools\serverpool \ ipdserver_creation_timeout = '120'.

The default value is 2 minutes (analogous with the COM instance creation timeout).

ServerTools-Client

- The server process 'UserExitPlugin.exe' is considered for displaying and killing PPR server processes

DPE 5.R17

- **Introduction** of exclusive client connections:
 - If a client connects exclusively the ipdserver will be reserved to be used only by this clients
 - If a server is used exclusively it will be tagged by a exclusive label to identify the user group connected to
 - If the exclusive label is unique no other client will be connected to the exclusiv labeled ipdserver process

PoolingServer

- The **configuration** will only be parsed and interpreted if the file date has changed - this reduces the process load since the configuration change is checked every 30 seconds the error logging for configuration parser has been improved by avoiding new error log entries for reading an unchanged configuration
- The reinitialization of the poolingserver remote connections will even be **proceeded** if a configuration failure has been detected and not immediately fixed by the user; before this improvement broken remote connections could not be reconnected as long as the poolingserver configuration were not parsed successfully.
- The configuration syntax has been extended to support a dedicated **serverpooling** for special groups of clients identified by an application or/and module label; clients using the EPIPDCClient to access the server will be automatically labeled with the exe name of the client process as the application label; to enable a dedicated client server pooling the client machine name in the poolingserver configuration has to be extended by the application and/or client module label.
- The automatic prioritization of machines has been removed:

Before PE5.17:

- If no priority were specified for a machine in the PoolingServer configuration, the priority was automatically set in the order of machine listing within the domain
 - to disable automatic prioritization the customer needs to manually set each machine priority - in most customer installations the right prioritiza-

tion is missing even with the default behaviour of automatically prioritizing the machines

After PE5.17:

- Since the automatic prioritization in the order of machine listing is not wanted to be used in a customer installation, it has been replaced by setting each machine with the highest priority until the priority is manually set in the configuration file
- **Asynchronous** initialization of the ServerTools process; this initialization includes the creation of the ServerPool and the startup of the load measurement via the Win PDH lib
- Since we found incidents on the customer (CITRIX) where the PDH initialization was delayed (5 minutes) the synchronous server tools initialization became a risk for the client connect to hang up resp. or delay a longer time
- If the PDH initialization fails by a timeout or hangs the user needs to be informed by an error message
- The client connect timeout is used for the server tools initialization too: HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\ ERGOPlan \ PoolingServer \ client_connect_timeout
- The **termination** of unused IPDServer processes is now asynchronously executed
- Since a hanging termination call within the ServerPool could lead to a blocked client login, the ServerPool call is proceeded asynchronously
- The use of write events for the IPDServer brokers, forcing the client login to **wait**, has been completely removed for the following common, mostly used pool management use cases :
- The PoolingServer configuration file has not changed
- The ServerPools does not needs to be reconnected
- The ServerPools only needs to be reconnected if the DCOM connection or the load measurements call backs do not respond this improvement reduces efficiently a large potential for waiting or hanging client logins; due to this enhancement the Broker cleanup time is reduced from 30 sec. to 20 sec. and the PoolingServer configuration change check time from 30 sec. to 15 sec.

ServerTools

- Periodical check (timeframe 10 seconds) if IPDServer processes are locked for **further** client connects (e.g. by the IPDServer emergency handler); in R16 this check was only proceeded within the connect routine, so a contemporary notification to the server tools clients about IPDServer lock state changes was not possible; since R17 the notification about the IPDServer lock state is displayed with a maximum delay of 10 seconds

ServerTools-Client

- **Displaying** the exclusive label and persistent state in the machine navigator tree

- Format: <machine name> [PID: <process id>, LABEL: '<exclusiv label>', persistent]
 - if the persistent flag is true <persistent> is displayed, otherwise not.
- Displaying information about the reason if an IPDServer is locked for **connect**:
- "out of memory" memory consumption has reached 2 GB minus the emergency failsafe reserved memory
- "critical exception" a critical exception has occurred (e.g. Access-Violation)
- "database exception" a critical database exception has occurred (e.g. database connection lost)
- "RPC connection lost" PRC connection to updatemng, lockmng or security logger lost
- "security log io" IO error while logging security data
- "terminating" IPDServer is being terminated by the IPDServer termination routine of the server pool
- "application" the IPDServer is locked for further connections using the ServerPool API

More information about the lock reason are written in the appropriate IPDServer error log

- The server process 'UserExitPlugin.exe' is considered for displaying and killing PPR server processes

DPE 5.17SP2

PoolingServer:

- **PoolingServer** configuration:
- Server domain options can be applied at machine level too
- Multiple server process emergency configurations can be defined and identified in the client-server net config by the emergency config name


```
<<emergencyconfig>>
      config_name = import_emergency
```

 - any emergency configuration can be referred at machine and domain level in the client server net config
 - in the clientnet the emergency configuration can extend an exclusive connect option; if the exclusive connect option is set in a client per API, it has to be marked by the new connect option 'connect_exclusive_api' e.g.:

```
client3 ( @opt = { connect_exclusive_api, emergency_config_name = import_emergency } )
```

Example servernet:

```
Location3 [server3 @opt = { emergency_config_name = import_emergency } ]
```
- The **emergency** configuration has been extended to support critical recurring error states

- If a critical error states recurses the system and the data consistency are in danger to be corrupted
- Before R17SP2 recurring error states has been handles only by an additional log entry to identify the state in the log
- Since R17SP2 the recurring error state can be configured in the PoolingServer configuration, section emergency configuration, to raise the abort emergency failover state, leading to closing all clients and their transactions

Example:

```
emergency_causes:
  critical_exception:
    abort [ recurring_error_state = { (20, 120, type_database), (20, 120,
type_c_exception), (20, 120, type_any) } ]
```

- The **emergency** configuration allows to configure an independent low memory handler by setting a treshold for the process virtual memory size. If the memory treshold is exceeded the memory emergency state is raised in the IPDServer process. The new low memory handler runs in an own thread in the IPDServer and measures the virtual memory size of its own process every 10 seconds (default).
- Configure handler param:
 - measurement interval
HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\IPDSERVER\ProcessMemoryMonitorIntervalSec
 - virtual memory threshold
HKEY_LOCAL_MACHINE\SOFTWARE\DELMIA\IPDSERVER\EmergencyThresholdAvailableVirtMemKB

If the memory threshold is set in the emergency configuration of the PoolingServer the registry value is ignored

Example:

```
emergency_causes:
  memory:
    abort [ process_available_virtual_kb = 10240 ]
    // default ipdserver memory emergency threshold = 10 MB
```

- **PoolingServer** Configuration
- The client net supports the configuration of client options (connect option, emergency option) for anonymous clients too based on the client program name identification (e.g. dpfframe.exe). Such an anonymous client configuration will be applied for that client program running on all machines as long as there is no machine specific entry overwriting the anonymous entry.

Example:

```
clientnet:
  @anonymous_userdomain [ @@dpfframe ( @opt = { connect_exclusive('E5 PPR-Navigator'), emergency_config_name = ppr_navigator_emergency } ) ]
```

DPE 5.18**Poolingserver**

- Main improvements to support medium and large DPE installation with far more than 3-4 IPDServer processes and heavy client short time connect frequency. Please refer to the [Load Balancing](#).

- The load balancing precision in R18 now does not depend anymore on the number of IPDServer processes running on a single machine. In previous Releases too large differences of the number of IPDServer processes between different machines could mislead the balancing algorithm. *Please refer to the [Load Balancing](#).*
- The short time balancing capability is increased by factor 20 from 20 seconds to 1 second. *Please refer to the [Load Balancing](#).*
- Improve calculation of the alternative 'availability' load index (disabled PDH measurement). *Please refer to the [Load Balancing](#).*
- The configuration parser has been extended in order to allow empty domains. *Please refer to the [The PoolingServer Configuration](#)*
- Logging of settings if they are changed compared with the setup defaults. *Please refer to the [PoolingServer Settings](#).*

ServerPool

- The termination of IPDServer processes in emergency failover mode has been improved to faster release bound resources (memory, database sessions, etc.).
- Introduce monitoring of the free available machine memory exhaust. *Please refer to the [PoolingServer Settings](#).*
- Logging of settings if they are changed compared with the setup defaults. *Please refer to the [PoolingServer Settings](#).*

ServerTools Client

- Display an error message in a monitor view if the connection to the monitor refreshing server process is lost (EPPoolingServer.exe or EPServerTools.exe).
- Display 'BiComm' / 'UniComm' connection mode hint in the IPDServer node text in the navigator view.
- New function to parse any PoolingServer configuration. *Please refer to the [Parse PoolingServer Configuration](#)*

DPE 5.19

Server Pool

- Global Emergency Mode is raised when a master process runs into a critical error state that affects functionality used by other processes. Critical errors are then resolved automatically, such that Process Engineer does not require a manual restart.

Global Emergency Mode saves the data consistency of pending data changes if one of the Master processes experiences a critical error state impacting central service functionality.

The following new Emergency state information is introduced with Global Emergency Mode functionality:

"Low memory of UpdateMng process on Master server"
 "Low memory of LockMng process on Master server"
 "Critical exception in UpdateMng process on Master Server"

"Critical exception in LockMng process on Master Server"
"Too many threads started in UpdateMng process on Master Server"
"Too many threads started in LockMng process on Master Server"
"Too many handles allocated in UpdateMng process on Master Server"
"Too many handles allocated in LockMng process on Master Server"

List of Figures

Figure 1: PoolingServer Diagram - Server Tools and Clients	4
Figure 2: Which Processes Runs Where.....	6
Figure 3: PoolingServer Configuration File.....	8
Figure 4: Client Labels	12
Figure 5: Client-Server Network.....	34
Figure 6: Starting Server Tools using the Start Menu.....	35
Figure 7: Interface when Starting the Server Tools	35
Figure 8: Connect PoolingServer, Connect ServerPool and Connect LockMng dialog box.....	36
Figure 9: Dialog Component Lock Information in which all Locked Transactions are Performed.....	36
Figure 10: Menu Bar	37
Figure 11: Server Tools with 3 Logged on Clients	38
Figure 12: PoolingServer and Server Tools with 2 logged on Clients	39
Figure 13: IPDServer process 1828 on PYRAMID:	40
Figure 14: IPDServer Process 1828 Locked	40
Figure 15: ServerPool Configuration	41
Figure 16: Error Message	41
Figure 17: ServerPool Resource Utilization.....	42
Figure 18: Resource Utilisation Monitor	42
Figure 19: Error Message	44
Figure 20: View will Contains an Error Message	44
Figure 21: Context Menu of the Pooling Server.....	45
Figure 22: Properties of the PoolingServer	45
Figure 23: Settings of the Pooling Servers Administration Client.....	46
Figure 24: "Show ServerPool Load Indexes" of the Pooling Server.....	47
Figure 25: PoolingServer Context Menu	48
Figure 26: PoolingServer configuration	48
Figure 27: Select File.....	48
Figure 28: Open Dialog.....	49
Figure 29: Error Message	49
Figure 30: Detailed Information	49
Figure 31: Last Parsed Text containing the Error Position.....	50
Figure 32: PoolingServer Configuration Completion.....	50
Figure 33: Terminate Server Processes.....	52
Figure 34: Context Menu of the ServerPool	52
Figure 35: Properties of the ServerPool	53

Figure 36: ServerTools Menubar	54
Figure 37: Show Process Information.....	54
Figure 38: Terminate IPD Server Processes.....	55
Figure 39: Server Tools Context Menu	55
Figure 40: The Server Registration Editor Settings	56
Figure 41: Track Process Memory	57
Figure 42: Context Menu on the Client.....	57
Figure 43: Termination Timeout	58
Figure 44: Note on Terminating the Client.....	59
Figure 45: Termination Process Message.....	60
Figure 46: Note on Terminating the Client.....	60
Figure 47: Message (to the client user) that the Client has been Terminated.....	61
Figure 48: Server Tools Message (to the administrator) after a Client has been Terminated	61
Figure 49: Login Dialog.....	63
Figure 50: Interactive Dialog	63
Figure 51: Information Dialog.....	63
Figure 52: Release Persistent Locks.....	64
Figure 53: Registry Entry Architecture	68

List of Tables

Table 1: Server Tools Process	33
Table 2: Threshold Values	42
Table 3: Registry Value Explanations	69
Table 4: Supported Values	70

Index

A

Apply button 60

C

ConfigFactoryCache 69

Configure the Server

Emergency behaviour 15

D

DCOM 7, 36, 62, 64, 73, 75

E

EPlogger 66

EPUnlockTool..... 62

Firewall..... 63

Persistent Locks..... 63

epunlocktool.exe 62

External Storage Files 66

F

Failover handling

Abort timeout 30

Close_transaction_timeout 30

Poet error 29

Firewall..... 63

H

Handles 65

Heap 65

L

Load Balancing

General 19

Load Index Calculation 19

Setting Changes R18 20

Lock PoolingServer 47

M

Main Storage..... 66

Master Server..... 4

Memory page fault..... 65

N

Nonliability ii

P

Pooling server..... 65

PoolingServer 4

Poolingserver Configuration File

Editing..... 9

Process ID 65

R

RAM 66

Registration Editor 7

Registry entry 66

Optional Registry Entry 68

Reload 37

S

Server Pool..... 26

Server Tools 66

clientlimit 26

Servernet..... 66

Serverpool 66

memorylimit..... 27

SERVERPOOL

Memorylimit..... 26

ServerPools

Context Menu

Settings 53

Subnet..... 66

T

terminate IPD Server Processes..... 50

Terminate IPD Server Processes..... 55

Terminate Processes..... 52

terminating clients

procedure:..... 58

Thread 66

Timeout 58

Timeout Extension..... 59

U

unlisted 15

Usage Data 66

Usage index 66

V

Virtual memory 66