GoMidjets – Automate your advantage

GoMidjets ClearEnv

Simple, fast and reliable software for central management of CM methodologies and IBM® Rational ClearCase® environments

Manual Guide

v1.0.5

Table of contents

Conventions Used in This Book 3
How to Contact Us 3
GoMidjets ClearEnv Concepts 4
Key features 5
GoMidjets ClearEnv installation 6
  Pre installation 6
  Server installation 6
  Clients installation 6
How to use GoMidjets ClearEnv 7
  As end-user 7
  As administrator / CM administrator 7
Appendix A: Methods 13
Appendix B: Naming Convention 16
Appendix C: Post scripts option 17
Appendix D: Technical information 17
Appendix E: ClearCase terminology 18

Conventions Used in This Book

Throughout this book, we've used the following typographic conventions:

Constant width
Constant width in body text indicates a language construct, such as the name of a stored procedure, a SQL statement, an enumeration, an intrinsic or
user-defined constant, a structure (i.e., a user-defined type), or an expression (like `dblElapTime = Timer - dblStartTime`). Code fragments and code examples appear exclusively in constant-width text. In syntax statements and prototypes, text set in constant width indicates such language elements as the function or procedure name and any invariable
elements required by the syntax.
Constant width italic
Constant width italic in body text indicates parameter names. In syntax statements or prototypes, constant width italic indicates replaceable parameters. In addition, constant width italic is used in body text to denote variables.
Italicized words in the text indicate intrinsic or user-defined function and procedure names. Many system *Italic*
elements, such as paths and filenames, are also italicized. URLs and email
addresses are italicized. Finally, italics are used for new terms where they are defined.

 This icon indicates a tip, suggestion, or general note.

 This icon indicates a warning or caution.

When “UNIX” is mentioned in this book, it includes all UNIX operating systems and all Linux operating systems that ClearCase can run on.

How to Contact Us

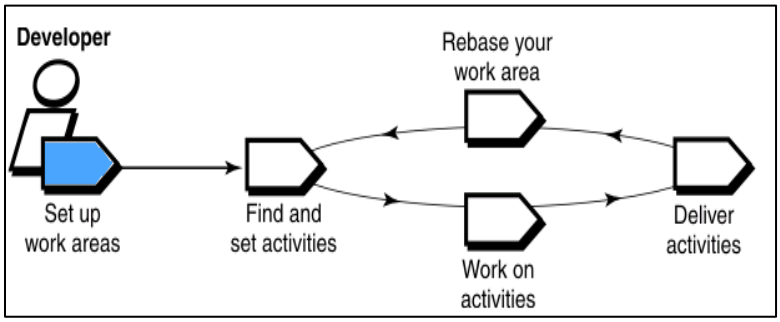
GoMidjets, Inc
68/15 Kaplansky
Petach-Tiqwa, Israel 49213
info@gomidjets.com
(972)-39212806 (fax)

GoMidjets ClearEnv Concepts

GoMidjets ClearEnv is intended to let the end-users easily create their own development environments. This target is achieved by configure methodologies that was pre-defined by CM administrator or organization management.

The end-users are developers, team-leaders, QA, technical writers and anyone who needs access to the ClearCase assets.

Actually, GoMidjets ClearEnv takes the “set up work areas” step as you may know from the ClearCase documentation, and promotes it beyond.



You may know the following chart, which is part of ClearCase documentation. You may know that “set up work areas” step is not as easy as it may look like. This step has a conceptual aspect – the methodologies - beside of the technical aspect, and it's a very critical step of a successful CM implementation process.

Technically, ClearEnv is installed on Windows host, and then it can serve all your environments: Windows, UNIX and Linux, both clients and servers.

Key features

- Automatic process to the end-users: Usually, the end-users just have to choose their project, and then GoMidjets ClearEnv makes all the rest.
- Suggesting 16 application development methodologies, including Agile, XP programming, Scrum, continuous integration. CMMI, RUP and refactoring.
- Support for parallel development and serial development.
- Support for ClearCase UCM versions starting from 2003.06, including ClearCase LT and ClearCase MultiSite ®.
- Support for all operating systems that run ClearCase, including Windows, UNIX and Linux.
- Support for cross-platform development of Windows-UNIX, Windows-Linux and UNIX-Linux.
- Unlimited customization - define projects as much as you need.
- Support for all NAS storage filers, including NetApp, EMC and so on.
- Support for Samba (SMB), NFS and so on.
- Support for local Windows views
- Support for UNIX and Windows view servers (by using storage locations)
- Support for dynamic and snapshot views
- Support for documentation projects
- Support for post scripts, including environment variables for much flexibility.
- Support for sub-VOB components
- Support for many UNIX regions (as much as needed)
- Support for creating environments by command-line.
- Support for naming convention of streams, views and storage locations
- Central management
- Full logs for each client, central management.
- Support for hierarchic UCM projects
- Integration to ClearCase Explorer
- Safety and security – The end-user has just to ensure that the whole parameters that were pre-defined for him are ok.
- Fully integrated to ClearCase UCM environment, including your scripts, your triggers and all other meta-data.
- Fully integrated to any 3rd party application, like ClearQuest®, ClearQuest UCM enabled, RequisitePro and so on.
- Immediately test your customized projects.
- Replace the Windows and UNIX "Join project" wizard. This wizard is usually confusing the end-users

GoMidjets ClearEnv installation

Pre installation

Pre-requisites:

1. A Windows host that runs ClearCase UCM (it can be ClearCase client or ClearCase server)
2. ClearEnv is based on your UCM environment. Please make sure that your CM system works properly: TCP/IP, VOBs and PVOBs, ClearCase registry server, license servers, filers, ClearCase clients and so on.

Server installation

Before the installation, make sure:
Administrative (local admin) privileges are required

Run ClearEnvServer.exe and follow the instructions, you will be asked to customize it also.

After the installation, you have to share the directory where the exe is installed.

Clients installation

Before the installation, please make sure the followings:

1. Administrative (local admin) privileges are required
2. The client installation does not contain any exe file – instead of it, it contains a shortcut to the exe file in the server. In hence, you have to share the directory where the server exe is installed.

Installation:
Client installations can be done in command-line mode only. This feature lets you the ability to install it remotely. Please consult your system administrator if necessary.

Open a cmd window and execute as following:

```
> cd installation-directory
> ClearEnvClient.exe /SILENT /PATH=\\hostname\ClearEnv-directory
```

e.g.:
If your ClearEnv server is named “server2003” and the exe is located in “ClearEnv” shared directory, you have to do the followings:

```
> cd c:\install
C:\install> ClearEnvClient.exe /SILENT /PATH=\\serrver2003\ClearEnv
```

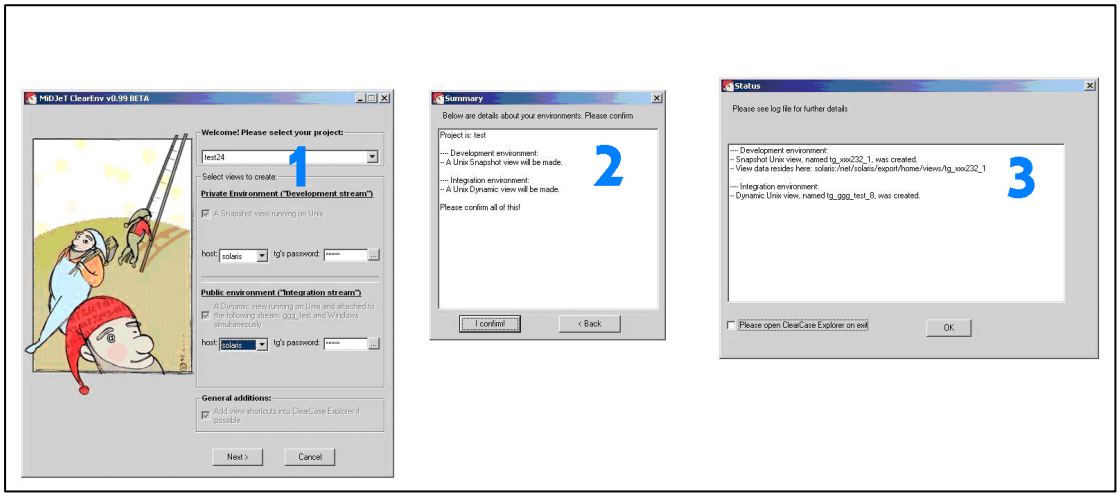
Do not omit the parameters. The installation will not work if they are omitted
You have to use capital and small letters as needed.

How to use GoMidjets ClearEnv

As end-user

This utility is intended to be smart & simple: choose your project and the utility will do most of the rest: suggest what environments to be built, what operating system and so on. You just have to approve all of this and see the results after it is done.

1. Choose product properties **2. Confirm details** **3. See results**

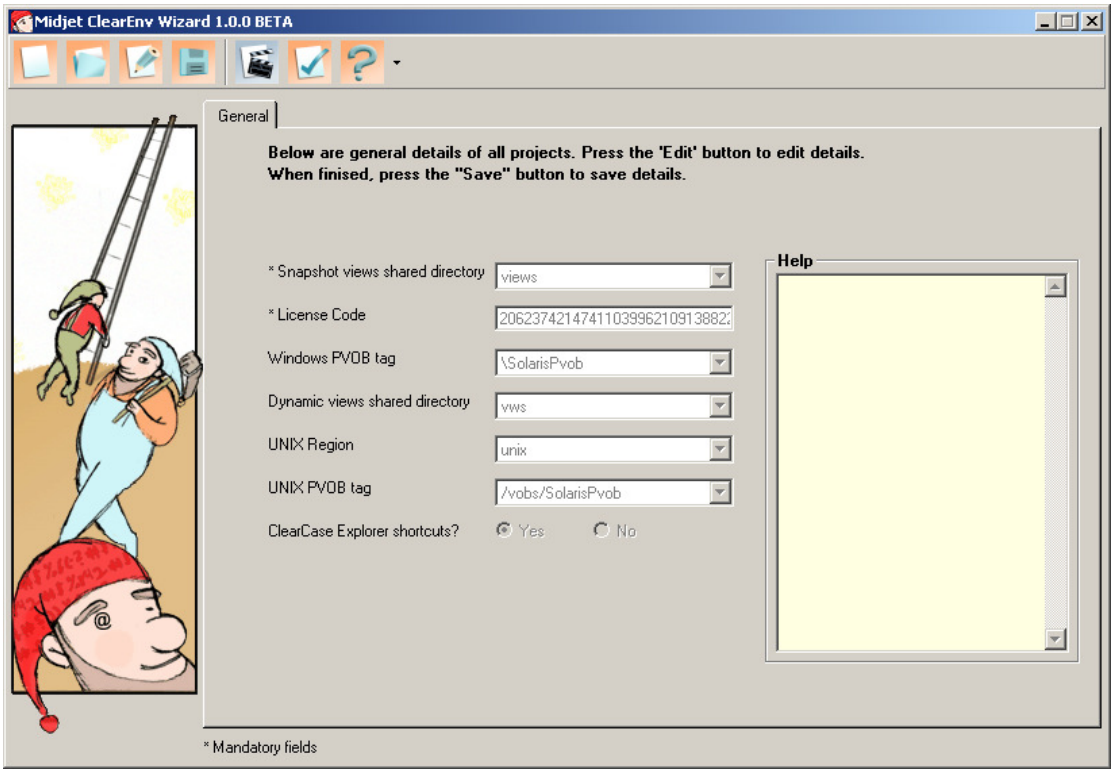


If you are going to make UNIX environments, another form might be shown between the first form and the second form. This form validates that all connections (accounts, TCP/IP and ClearCase) to UNIX host are working properly.

As administrator / CM administrator

As administrator, you have to customize the projects and parameters of your organization. Don't worry – It takes a few minutes and it is done automatically as much as possible.

When you install the ClearEnv server, you will be asked if you want to start configuring your organization properties.



ClearEnv Wizard – General properties

Toolbar

Toolbar buttons are:
New product – Create a new product
Load product – Load an exist product
Edit product – Edit an exist product. You have to press this button if you want to make a change to the product properties
Save product – Save the product properties
Test product – Validate product properties and execute the end-user utility as a test
Validate product – Validate product properties
Help – General help and this manual

General tab

The section contains two kinds of parameters: mandatory and optional parameters. It depends also if you have UNIX or Linux environments or not (only Windows environments).

There are two mandatory fields: Snapshot views shared directory and License code.

Snapshot views shared directory - the shared directory where all snapshot views information is reside in. This directory **must** be identical on all Windows clients.

 This directory is part of the naming convention of your organization.

License Code – License information. This parameter is mandatory. The utility will not run unless the license if valid.

The general optional fields for Windows are:

PVOB – Name of Windows PVOB (Project VOB) tag of all projects. If you have more that one PVOB, you can define it separately to each product (see below). The PVOB convention is Windows-like, so you have to specify "\" sign.

Dynamic views shared directory – the shared directory where all dynamic views information is reside in. This directory **must** be identical on all Windows clients.

 This directory is part of the naming convention of your organization.

Password – This feature is added by a request of customers who own a Unix VOB server while most of their developers are working on Windows environments. Since ClearCase forces them to open UNIX accounts to each of the users and the users do not use it interactively, so they prefer to use unsafe "generic" password. If you use this parameter, its value will be the default password for all users, so they will not have to enter the UNIX password when they make new environments.

 Using this parameter is not so recommended.

Force Snapshot – In some cases, you may want to override a rule and enforce your users to use snapshot files, even if usually you prefer them to use dynamic view. This parameter is intended exactly to such situations!
For example: You define that the integration view will be dynamic, but some of your users own a laptop, and they prefer to "take" the integration view home and use it at home. If you enforce
Is valid on an evaluation version???

Prompt – if you have any kind of at least one Unix view, you may have to define this parameter. This parameter defines the UNIX or Linux prompt and it's used to run UNIX commands. If not set, the prompt is "….."

You can use regular expressions so set this parameter.
Read here for further details about regular expressions: <put some web link here>

The general optional fields for UNIX and Linux are:

PVOB – Name of Windows PVOB (Project VOB) tag of all projects. If you have more that one PVOB, you can define it separately to each product (see below). The PVOB convention is Windows-like, so you have to specify "\" sign.
The product PVOB, if exists, will be read first.

UNIX PVOB tag – A UNIX convention of the PVOB tag. For example: /vobs/myvob

UNIX Region – Name of UNIX region.

Dynamic Views Shared directory – the shared directory where all dynamic views information is reside in. This directory **must** be identical on all Windows clients (It's part of the naming convention!)

Password – This feature is added by a request of customers who own a Unix VOB server while most of their developers are working on Windows environments. Since ClearCase forces them to open Unix accounts to each of the users and the users do not use it interactively, so they prefer to use unsafe "generic" password. If you use this parameter, its value will be the default password for all users, so they will not have to enter the UNIX password when they make new environments.

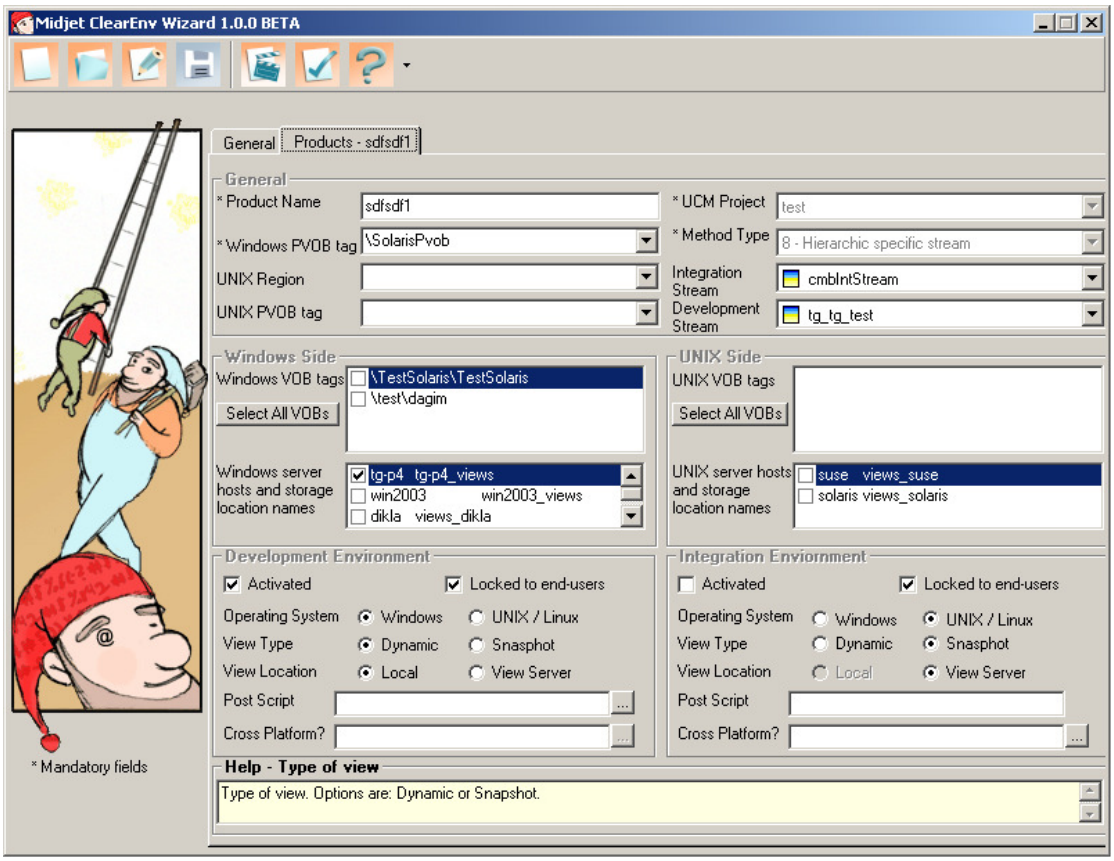
 Using this field is unsecured and in hence is not so recommended.

Force Snapshot – In some cases, you may want to override a rule and enforce your users to use snapshot files, even if usually you prefer them to to use dynamic view. This parameter is intended exactly to such situations!
For example: You define that the integration view will be dynamic, but some of your users own a laptop, and they prefer to "take" the integration view home and use it at home. If you enforce
Is valid on an evaluation version???

Prompt – if you have any kind of at least one Unix view, you may have to define this parameter. This parameter defines the UNIX or Linux prompt and it's used to run UNIX commands. If not set, the prompt is "….."

You can use regular expressions so set this parameter.
Read here for further details about regular expressions: <put some web link here>

Products tab



ClearEnv Wizard – A product properties

Open new product:

The following fields are being inheritance from the general properties:
Windows PVOB tag
UNIX PVOB tag
UNIX region
UNIX server hosts and storage location names

The meaning is that when you open a new product, the default value will be as in the general properties, but you can change it and set the specific parameters per product.

In this section, you can define separately properties of each "product".

A product is based on a ClearCase UCM project, and it includes all the properties around.

As the “General” tab, this tab contains also mandatory and optional parameters, and it depends if you have UNIX or Linux environments, or not.

The mandatory parameters for Windows are:

Product Name - Here you define the product name, as it is seen when your end users have to choose the product.

Method – Here you specify the method number. Please see below for the options.

Windows PVOB tag –

UCM Project – Name of ClearCase UCM project.
This parameter is case sensitive. Do not include the PVOB name in this field.

Optional fields are:

Windows VOB tags – Name of VOB tags to be mounted or loaded. If you specify dynamic view in your product, so here you specify the VOB names to be mounted. If you specify snapshot views in your product, so here you specify the VOB names to be loaded.
This parameter is case sensitive. You can specify here more that one VOB. In this case, the VOB names have to be separated by semicolon (;) character.

Development Environment frame – This frame includes all development view properties:
Activated –
If you want that a development view will be made when the end-user runs this application, check this box.
Operating System – What operating system the view will be made on. Options are: Windows or UNIX.
View Type – Type of view. Options are: Dynamic or Snapshot.
Post Script – Here you can specify a script name that will be run after the view is made successfully. You have to specify here the script path as it is run by appropriate user account (local account for Windows view; the user that was defined if it is a UNIX view).
If it’s a UNIX view and you’ve specified the “CrossPlatform” path, the script will be run in the UNIX host.

 Please see appendix C for further details and examples.

Cross-Platform – This option is valid in case your operating system is UNIX and the view type is dynamic. This option means that you can set the equivalent Windows view tag. If you want to set it, you have to specify here the appropriate path of the Unix view. For example: If your Unix view are stored in foobar:/export/home/views , what means it is stored in *foobar* machine, in /export/home/views file system path, you have to specify the equivalent Windows path as is defined by using Samba / NFS / NetAPP cifs / EMC .
Take into consideration that you have to verify that the connection works and all relevant permissions are working well.

 This option is not valid on ClearEnv version for ClearCase LT.

Tmode – FFU.
Locked to end-users – Check this box if you do not want to let the end-users the option to choose if a development view will be installed or not. If you check this box, this option will be read-only to end-users.
In fact, using this checkbox with a combination of the “Development Environment” checkbox lets us 4 combinations:

 By using combinations of “Activated” and “Locked for end-users” field, you have 4 combinations:
1 – Activated and non-locked: The view is checked and enabled, what means that this view will be made, but the end-user can cancel its creation.
This state is the default behavior if omitted.
2 – Non-activated and non-locked: The view is unchecked and enabled, what means that this view will not be made unless the user mark the view checkbox
3 – Activated and locked: The view is checked and disabled, what means that this view will be made and the end-user cannot cancel its creation.
4 – Non-activated and locked: The view is unchecked and disabled, what means that this view will not be made and user cannot change it.

 You cannot select non-activated and locked option for both development and integration environments for the same product (Think about it – it means that no environment will be made at all).

The mandatory parameters for UNIX or Linux environments are:

Product Name - Here you define the product name, as it is seen when your end users have to choose the product.

Method – Here you specify the method number. Please see below for the options.

Unix VOBs – List of Unix VOBs to be loaded or mounted.
Unix Locations – This is a list of pairs: hostname and storage location. The hostname is the UNIX (or Linux) host where the view(s) will reside in, and the storage_location is the name of the ClearCase storage location.
If you want to let your users choose a host from a list, check all necessary hosts.

 You should use ClearCase storage location if you want to make any unix views. Read in ClearCase manual for further details.

Optional parameters are:

PVOB – Please specify the PVOB (Project VOB) here if the project Product PVOB is different of the general PVOB.
This parameter is case sensitive.

IntView – This line contains all relevant parameters of integration environment. Its parameters are identical to "DevView" parameters (see above).

CCExpShortcut – Do you want that ClearCase Explorer shortcuts will be made automatically for you?
Values are: True and False.

The following parameters are relevant to methods 4 to 8:
Integration Stream – Name of parent (integration) stream for this product. In opposite to the first four methods, where the integration stream is the real integration stream of the project, if you specify this parameters, it concerns to hierarchic project where its internal stream that can be a child or a grandson of the real integration stream.
This parameter is case sensitive. Do not include the PVOB name in this field.

Development Stream – Name of the specific development stream to be based on. It is relevant to methods #4 and #8. In these methods, you have to specify which stream will be considered your development stream.
This parameter is case sensitive. Do not include the PVOB name in this field.

☞ When you save the product details, a validation check is automatically done before. You cannot save details unless the validation is passed successfully.

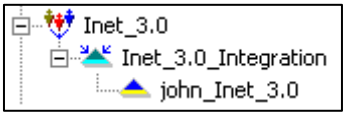
☞ When you test a product, it is automatically saved (and validates) before the test is being started.

Appendix A: Methods

Since there are many methodologies implemented in the application, we have donates this part to describe it. The methods options are:

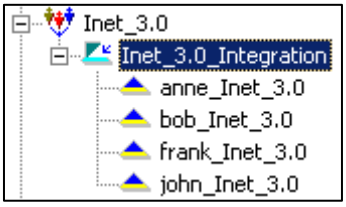
1 – Stream per developer
A development (private) stream will be made for each developer, by its **login** name, and then a view will attached to this stream.

For example: If I login by “john” account and the project name is “Inet_3.0”, so a new stream named “john_Inet_3.0” will be made, and then a new view will be made also.
Seeing it with ClearCase Project Explorer, it will be seen like the following chart:



If this private stream is already exist, so the utility identifies it, notifies the end-user that this stream is already exist and it contains one view or more, and asks him if he wants to create a new view.

Another example: If your project contains 4 developers: Anne, Bob, John and Frank, and you want a separated stream for each of them, you can define the project by using this method. After each of them run this application from their own machine, your project will be seen like the following chart:



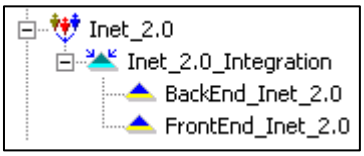
A recommendation: Use snapshot-view developer workspaces. If developers are to work on a private stream, then dynamic views are not really an option. The power of dynamic views is that they update themselves automatically when the branch that they are looking on is updated.

Summary

Useful usages: Conceptually, this method appropriates to projects where you need an isolation of check-in and check-out for each user.

Useful usages	Conceptually, this method appropriates to projects when you nee an isolation of check-in and check-out for each user.
Well-known methodologies	

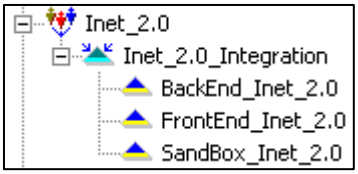
2 – Choose stream from a list
A view will be made, attached to existing streams (No new stream will be made).
For example: In project “Inet_2.0” all the developers are working on two streams only: BackEnd and FrontEnd (It can be that there are 2 teams in this project, and each of the developers belongs to one team or more).
When the end-user run ClearEnv and choose Inet_2.0 product, he will be asked about one of the two existing streams. After he chooses one, a new view, attached to the chosen stream, will be made.



Summary

Useful usages	
Well-known methodologies	Agile, SCRUM

3 – A view will be made, attached to existing streams, but the end-user has an option to create a new stream.
For example: In project “Inet_2.0” all the developers are currently working on two streams: BackEnd and FrontEnd.
Now, one of the developers has been asked by his team leader to try a new feature on an isolated environment, just as a “sandbox”. So this developer has the ability to run ClearEnv and chooses Inet_2.0 product, and then he can choose one of the two existing streams, or just create a new stream and a view attached to it. Let’s say that the developer has chosen to make a new stream which called “sandbox_Inet_2.0”. So at the end of ClearEnv running, it will be looked like the following chart:



Summary

Useful usages	
Well-known methodologies	Agile, SCRUM

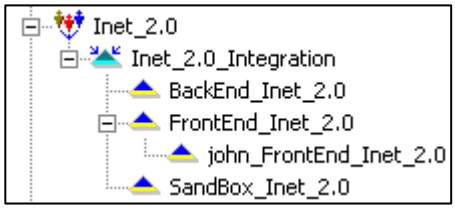
4 – A view will be made, attached to existing and pre-defined stream (no new stream will be made).
For example: In project “Inet_2.0” we have 3 development streams: BackEnd, FrontEnd and SandBox (just as the previous chart), and currently we want that every new developer that joins the project and have to make a new view, will be attached to “FrontEnd” stream.
Now, in addition to setting the method parameter in the configuration file, we have to specify another parameter: Development stream.

Summary

Useful usages	
Well-known methodologies	Agile, SCRUM

5 – This method is quite similar to method #1. As in #1, a development (private) stream will be made for each developer, by its **login** name, and a view will attached to this stream. The difference is that the new stream will not be based on the integration stream, but on another stream in this project that will be considered as an integration stream.

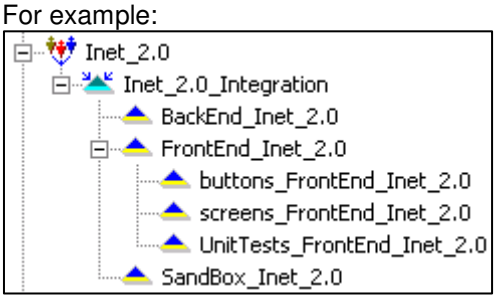
For example: If I login by "john", the project name is "Inet_2.0", and the parent stream to be considered as integration stream is called "FrontEnd_Inet_2.0", so a new stream named "john_Inet_3.0" will be made **under "FrontEnd_Inet_2.0"** stream, and then a new view will be made also.
Seeing it with ClearCase Project Explorer, it will be seen like the following chart:



Now, in addition to setting the method parameter in the configuration file, we have to specify another parameter: Integration stream.

 It is recommended to recommend the baseline on the stream that is considered as the integration stream, since the new stream is based on the parent recommended baselines.

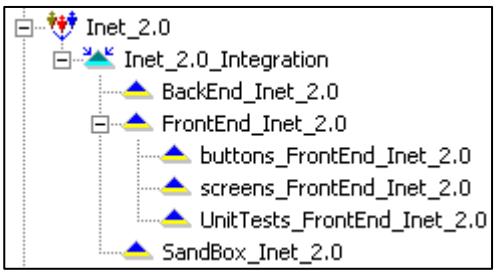
6 – This method is quite similar to method #2. As in #2, A view will be made, attached to existing streams (No new stream will be made). The difference is that the new stream will not be based on the integration stream, but on another stream in this project that will be considered as an integration stream.
For example:



In project "Inet_2.0" we have a development stream named "FrontEnd_Inet_2.0" that has child streams, so it considered also as integration stream for his children. We want that all the developers will work on its child streams (buttons, screens or UnitTests). When the end-user run ClearEnv and choose Inet_2.0 product, he will be asked about one of the three existing streams. After he chooses one, a new view, attached to the chosen stream, will be made.
Now, in addition to setting the method parameter in the configuration file, we have to specify another parameter: Integration stream.

Some useful usages:

7 – This method is quite similar to method #3. As in #3, a view will be made, attached to existing streams, but the end-user has the option to create another stream. The difference is that the new stream will not be based on the integration stream, but on another stream in this project that will be considered as an integration stream.
For example:

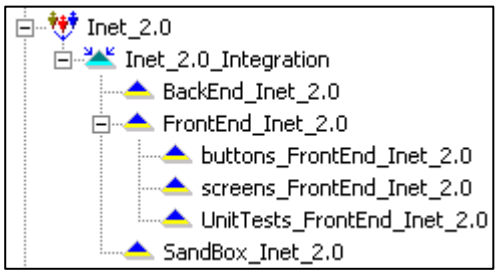


In project "Inet_2.0" we have a development stream named "FrontEnd_Inet_2.0" that has child streams, so it considered also as integration stream for his children. We want that all the developers will work on its child streams ("buttons", "screens" and "UnitTests"). When the end-user run ClearEnv and choose Inet_2.0 product, he will be asked about one of the three existing streams. After he chooses one, a new view, attached to the chosen stream, will be made.

Now, in addition to setting the method parameter in the configuration file, we have to specify another parameter: Integration stream.

In fact, you can see this method as a combination of methods #3 and #6.
You cannot choose a stream to be considered as an integration stream if this stream does not have any child stream.

8 – This method is quite similar to method #4. As in #4, A view will be made, attached to existing and pre-defined stream (no new stream will be made). The difference is that this pre-defined stream does not be based on the integration stream, but on another stream in this project that is considered as an integration stream.
For example:



In project "Inet_2.0" we have 3 development streams: BackEnd, FrontEnd and UnitTests, and currently we want that every new developer that joins the project and has to make a new view, will be attached to "UnitTests" stream.
Now, in addition to setting the method parameter in the configuration file, we have to specify also two parameters: Development stream and Integration stream.

You cannot choose streams if they do not have parent-child relation.

The same behavior if the view is made on a UNIX machine.

Appendix B: Naming Convention

ClearEnv applications keeps **naming convention** of streams, views and storage locations

Views convention –
username_streamname (all methods!)

Streams –
You can choose from the following:
username_projectname (method #1)
pre-defined branch
any naming you want

Hierarchic-nested branch:

As a CM administrator, you can make the streams by yourself, and then force the end-users to use the existing streams only.

if the stream starts with the username, so the utility identifies it and do not duplicate the username!

You can force the end-users to use pre-defined storage locations, and they cannot change it!

Appendix C: Post scripts option

As written above, you can specify your own script name that will be run after the view is made successfully. This feature empowers the automation process that can be done by ClearEnv. Usages might be:

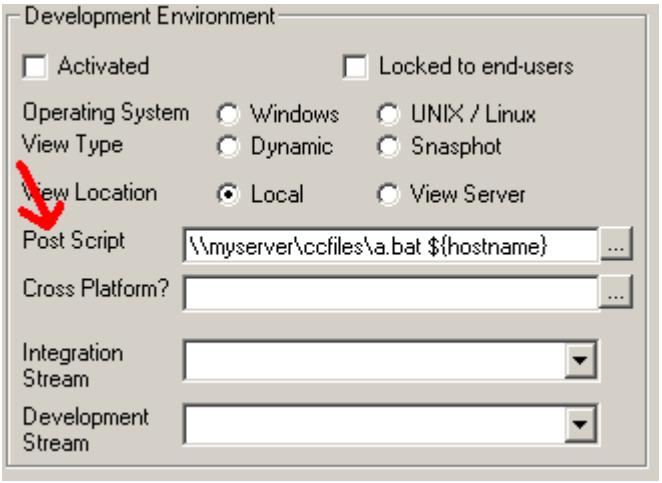
1. Sending a mail to the CM administrator, containing details of the new created environment
2. Creating a new activity on the new environment (stream and view)
3. Copying files into the new environment (view)
4. Checking out some files
5. Any other needs of your organization.

The script can be on any programming language that appropriate to relevant operating system. In Windows, it can be batch files, shell command-lines, Perl, VB script, WSH (power scripts) and so on.
In UNIX and Linux, it can be shell scripts and command-lines, Perl and so on.

For your convenient, GoMidjets ClearEnv supplies environment variables that can help you empower the script by using them as arguments. The variables are:
${view} – Final path and tag name of the new created view
${username} – Username of the end-user who ran ClearEnv
${hostname} – Hostname of where ClearEnv was being running from.
${stream} – Stream name of the new created environment

Examples:

A. The following example demonstrates how to send the hostname as argument to another script.
“a.bat” script is a batch script that will considered the first parameter as hostname.



Here is the content of “a.bat” script:

```
C:\ccfiles> type a.bat
echo views made in %1 on %date%
```

Appendix D: Technical information

How is this product working?

- Validation checks according to the arguments that the end-user is entered
- Create a new stream if necessary
- Create a new view (depends on view type, operating system and many other factors)
- Load rules if it's a snapshot view; otherwise – mount VOBs.
- Setting tags if necessary (cross-platform)
- Running post script if necessary
- Create ClearCase Explorer shortcuts if necessary

During this process, the log file is filled automatically. At the end of the process there is a log file and an output display to the end-user.

Appendix E: ClearCase terminology

Please see here.