This document contains information on the operation of the SHI Software on the IDC Development LAN, including baseline descriptions and monitoring, troubleshooting and change procedures.

Summary

This document describes the procedures used to operate and maintain the Seismic, Hydro-acoustic and Infrasound (SHI) processing software on the IDC Development LAN. It also describes the hardware, software and environment baseline, which is needed to properly operate the Development LAN software. The chapters on the Development LAN baseline and on change procedures are of general interest for all Development LAN users and developers.
## Document history

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>7 June 2004</td>
<td>Alexander Boresch</td>
<td>draft document structure</td>
</tr>
<tr>
<td>0.2</td>
<td>12 August 2004</td>
<td>Marian Harustak</td>
<td>partial draft procedures in chapters 3 and 4</td>
</tr>
<tr>
<td>0.3</td>
<td>16 September 2004</td>
<td>Alexander Boresch</td>
<td>complete draft version</td>
</tr>
<tr>
<td>0.4</td>
<td>15 October 2004</td>
<td>Alexander Boresch</td>
<td>revised draft version</td>
</tr>
<tr>
<td>1.0</td>
<td>28 November 2004</td>
<td>Alexander Boresch</td>
<td>final SHI version</td>
</tr>
<tr>
<td>1.1</td>
<td>16 August 2005</td>
<td>Alexander Boresch</td>
<td>baseline for expanded version (SHI, RN, ATM)</td>
</tr>
<tr>
<td>1.2</td>
<td>7 February 2006</td>
<td>Henrik Edlund</td>
<td>ATM chapters added</td>
</tr>
<tr>
<td>1.3</td>
<td>14 December 2006</td>
<td>Gertrud Waich</td>
<td>added autoconf build procedures, updated SHI chapters</td>
</tr>
<tr>
<td>2.0</td>
<td>5 January 2007</td>
<td>Gertrud Waich,</td>
<td>full SHI, RN and ATM version</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Alexander Boresch</td>
<td></td>
</tr>
<tr>
<td>3.0</td>
<td>20 May 2008</td>
<td>Vladimir Gelashvili,</td>
<td>document re-structured, updated and split into separate volumes for SHI, RN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Alexander Boresch,</td>
<td>and ATM software</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Jan Wüster (review)</td>
<td></td>
</tr>
</tbody>
</table>
# Table of Contents

1. **Introduction** ........................................................................................................... 6  
   1.1. Identification and Purpose of the Document .......................................................... 6  
   1.2. Purpose and Role of the Development LAN ............................................................ 6  
   1.3. Document Overview ................................................................................................ 7  
   1.4. Typographical Conventions ..................................................................................... 7  

2. **General Baselines and Procedures for all Technologies** ........................................... 8  
   2.1. Roles and Responsibilities ....................................................................................... 8  
   2.2. Generalized Hardware Description ......................................................................... 9  
   2.3. Generalized Infrastructure Software Description .................................................. 10  
   2.4. Directory Structure .................................................................................................. 10  
      2.4.1. Top-level Directory ........................................................................................... 10  
      2.4.2. Technology-Specific Directories .................................................................... 10  
   2.5. **ClearCase** Baseline, Configuration Management and Build Procedures .............. 11  
      2.5.1. ClearCase Terminology and Role at IDC .......................................................... 11  
      2.5.2. ClearCase VOBs ............................................................................................. 12  
      2.5.3. IDC Software Development Cycle ................................................................... 13  
      2.5.4. ClearCase Build and Installation Procedures ..................................................... 14  
      2.5.5. More on autotools ............................................................................................ 16  
   2.6. Change Request and Problem Reporting Procedures ............................................. 19  
      2.6.1. Introduction ...................................................................................................... 19  
      2.6.2. Requesting and Implementing Changes on the Development LAN ................. 19  
      2.6.3. Changes to the Baseline Inventory ................................................................... 20  
      2.6.4. Changes to Development LAN Procedures ..................................................... 21  
      2.6.5. Problem Reporting .......................................................................................... 21  

3. **SHI data acquisition and processing** ........................................................................ 22  
   3.1. Introduction .............................................................................................................. 22  
   3.2. Baseline Inventory ................................................................................................... 22  
      3.2.1. Hardware Baseline .......................................................................................... 22  
      3.2.2. Application Software Baseline ....................................................................... 23  
      3.2.3. Software Location ........................................................................................... 24  
      3.2.4. Software Configuration Baseline ..................................................................... 24  
      3.2.5. Directory Structure .......................................................................................... 25  
      3.2.6. SHI User Environment .................................................................................... 28  
      3.2.7. Data Acquisition and Processing .................................................................... 30  
   3.3. Operations Procedures ............................................................................................. 37  
      3.3.1. Starting the SHI Processing System .................................................................. 37  
      3.3.2. Starting the CDTools and cd2w ...................................................................... 37  
      3.3.3. Starting the Message and Subscription Subsystems ......................................... 38  
      3.3.4. Starting the DACS ......................................................................................... 39  
      3.3.5. Starting the Interactive Analysis Software ........................................................ 40  
      3.3.6. Starting the Web Subsystem ............................................................................ 41  
      3.3.7. Shutdown of the SHI Processing System ............................................................ 41  
      3.3.8. Shutdown of the DACS ................................................................................... 42  
      3.3.9. Shutdown of the Message and Subscription Subsystems .................................. 43  
      3.3.10. Shutdown of the CDTools and cd2w ................................................................. 43  
      3.3.11. Shutdown of the Interactive Analysis Software ................................................. 44  
      3.3.12. Shutdown of the Web Subsystem .................................................................... 44  
      3.3.13. Starting and Stopping Cron Jobs ..................................................................... 45
3.3.14. Clean-up Cron Jobs ................................................................. 45
3.3.15. Controlling Data Migration and Purging ............................... 46
3.3.16. Controlling Data Archiving .................................................... 47
3.3.17. Controlling keep_alive Scripts ............................................... 47
3.3.18. Controlling Web Subsystem Cron Jobs ..................................... 48
3.3.19. Controlling Other Cron Jobs .................................................. 48

3.4. Monitoring Procedures ................................................................ 50
3.4.1. Routine Monitoring Tasks ........................................................ 50
3.4.2. Monitoring the Automatic Processing WorkFlow ....................... 50
3.4.3. Monitoring the Data Request WorkFlow ..................................... 52
3.4.4. Monitoring the Data Migration WorkFlow ................................. 53
3.4.5. Monitoring the Data Archiving WorkFlow ................................ 54
3.4.6. Monitoring the CDTools and cd2w ............................................ 55
3.4.7. Monitoring the DACS ............................................................... 56
3.4.8. Monitoring Message and Subscription Subsystem Processes ........ 57
3.4.9. Monitoring the Processing Status in the Database ...................... 58
3.4.10. IDCX.interval Table ............................................................. 59
3.4.11. IDCX.request Table .............................................................. 60
3.4.12. IDCX.msgdisc Table ............................................................. 60
3.4.13. IDCX.wfdisc Table ............................................................... 61
3.4.14. Other Relevant Tables .......................................................... 62

3.5. Maintenance Procedures ............................................................ 64
3.5.1. Installation of New SHI Stations ............................................... 64
3.5.2. Tuxedo Configuration on a New Server ..................................... 65
3.5.3. Synchronizing the Development LAN Configuration with the Testbed LAN 66

3.6. Troubleshooting Procedures ....................................................... 73
3.6.1. Log File Location for Different Applications .......................... 73
3.6.2. Routine Checks and Monitoring .............................................. 73
3.6.3. Interpreting Error Messages and Resolving Problems .............. 74
3.6.4. Solving Common Problems .................................................... 75
3.6.5. Analysis and Logging of Unknown Problems .......................... 78

Terminology ....................................................................................... 80
Glossary ......................................................................................... 80
Abbreviations ................................................................................ 83
References ...................................................................................... 84
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>IDC technologies and data processing systems</td>
<td>6</td>
</tr>
<tr>
<td>Figure 2</td>
<td><em>autotools</em> build procedure</td>
<td>17</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Subsystems of SHI data processing</td>
<td>31</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Components of station processing</td>
<td>33</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Components of network processing</td>
<td>34</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Auxiliary data retrieval path on the IDC Development LAN</td>
<td>35</td>
</tr>
</tbody>
</table>
1. INTRODUCTION

1.1. Identification and Purpose of the Document

This document describes the procedures used to operate and maintain the SHI processing software on the IDC Development LAN. It provides an overview over the existing infrastructure and also describes the hardware and software environments, which are needed to properly operate the Development LAN software. The addressed audience are primarily the Development LAN operators, but also includes developers, testers, users and administrators of the Development LAN hardware and software.

There are two companion documents, [IDC/DEVLAN/OG-RN] and [IDC/DEVLAN/OG-ATM], which describe the Development LAN’s Operations and Maintenance procedures for the Radionuclide processing software and the Atmospheric Transport modelling software, respectively.

The current document is based on the general framework provided by the following documents:

- IDC Operational Manual, [IDC-OM]: Describes the general operational framework and mission of the IDC. Available at H:\Conference Documents\Official Documents\PMOs and AG\WGA\WGB\Operational Manuals\IDC OM on the CTBTO Intranet.

- Draft Procedures of the IDC Configuration Control Board (CCB), [IDC-CCB]: Describes the software and configuration change procedures for the IDC Testbed and Operations LANs. Available at: http://idc030.idc.ctbto.org:8000/Docs/CCB-Documentation.html in the CTBTO Intranet.


1.2. Purpose and Role of the Development LAN

The Development LAN was created in July 2002 and serves as a development and test platform for the IDC software. This includes processing software for all CTBT monitoring technologies Seismic, Hydro-acoustic and Infrasound (SHI) data processing, Radionuclide (RN) data processing and Atmospheric Transport Modeling (ATM).

![Figure 1 IDC technologies and data processing systems](image-url)
The Development LAN is owned by the IDC Software Integration unit and is under less rigorous configuration control than other IDC LANs (Testbed and Operations LANs). When new or modified software becomes available and has successfully passed unit testing by the developers, it is installed on the Development LAN to be tested in the integrated IDC data processing system for the relevant technology. Such integration testing checks if the new or modified software component functions well within the framework of the whole SHI processing system and helps to identify any unintended effects of the installed software change on other processing software. If such effects are found, they are analyzed and resolved before the software change is implemented on the Testbed LAN for operational testing, and finally on the Operations LAN for use in regular IDC operations. Thus the Development LAN plays a major quality assurance role in the IDC software development cycle.

### 1.3. Document Overview

Chapter 1 gives an introduction to this document.

Chapter 2 contains general information which is common for all processing technologies. It is organized in sections describing roles and responsibilities, the hardware and infrastructure baselines, the general directory structure, the role and use of the configuration management system *ClearCase* at the IDC, and generic change procedures. The information in this chapter is of general interest for all Development LAN users.

Chapter 3 contains all SHI-specific information including the baseline inventory and procedures for routine operations, system monitoring, system maintenance and troubleshooting. The primary audience for this chapter are the operators of the SHI processing software. However, the baseline information and troubleshooting procedures may also be of interest for developers and testers.

### 1.4. Typographical Conventions

<table>
<thead>
<tr>
<th>Element</th>
<th>Font</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>application, process, programming language</td>
<td><em>italic</em></td>
<td><em>Workflow, Perl</em></td>
</tr>
<tr>
<td>database account, database table, database attribute</td>
<td><strong>UPPERCASE. bold. normal</strong></td>
<td><strong>IDCX. interval. intvlid</strong></td>
</tr>
<tr>
<td>configuration parameter, variable, literal value, command, computer code, machine name</td>
<td><em>Courier New 11</em></td>
<td><em>ssh cmss@&lt;machine&gt;, $(CMS_CONFIG)</em></td>
</tr>
</tbody>
</table>
2. GENERAL BASELINES AND PROCEDURES FOR ALL TECHNOLOGIES

2.1. Roles and Responsibilities

The following terms are used in this document to designate users, operators, administrators and developers of the Development LAN and its related infrastructure:

**ClearCase Administrator**: Maintains ClearCase including all VOBs and views described in section 2.5. Owns the UNIX account vobadm. To be contacted for all ClearCase issues and questions.

**ClearQuest Administrator**: Maintains ClearQuest. ClearQuest is used to report all software problems at the IDC (see section 2.6.5.).

**Data Recipient**: A recipient of continuous waveform data forwarded in either CD-1.0 or CD-1.1 format from the Development LAN. Data recipients can be internal within the IDC (another Dev LAN machine, Services Unit, IMS, etc.) or external (NDCs, other organizations) and are identified by a name and an IP address. The technical point of contact for each data recipient should be contacted for all issues and changes related to the data recipient.

**Data Provider**: A sender of continuous waveform data in either CD-1.0 or CD-1.1 format received by the CDTools on the Development LAN, or a sender of auxiliary waveform data or radionuclide data received by the Message Subsystem. Data providers can be internal (Operations LAN, TestBed LAN, IMS) or external (stations, NDCs) and are identified either by one or more IP addresses and associated station names (for continuous data), or by one or more email addresses (for auxiliary waveform data and radionuclide data). The technical point of contact for each data provider should be contacted for all issues and changes related to the data provider.

**Database Administrator**: Operates and maintains the IDC database servers and instances. To be contacted for all database issues that cannot be resolved by the Development LAN operator or the individual user.

**Development LAN Operator**: Operates and maintains the Development LAN data processing system for one or more technologies including all application software and configuration, LAN-specific processes, the system-specific directory structure, the generic user environment and the associated baseline inventories described in section 3.2. To be contacted by users who wish to modify any of these baselines, install new application software or plan to perform software tests on the Development LAN. See section 2.6.2. for requesting software changes on the Development LAN.

**Development LAN User**: Anyone who wishes to work on the Development LAN and does not fall into any of the other defined roles. Typical Development LAN users are IDC software developers and testers who use the Development LAN to improve and test the IDC software to be later installed on the other IDC LANs. Users need a personal UNIX/Linux account on the Development LAN, which is created by the system administrator.
Processing Engineer: Operates and maintains the IDC software on the Operations and Testbed LANs. Technical point of contact if one of these LANs acts as data provider or recipient for the Development LAN.

Web Administrator: Operates and maintains the IDC web servers including the web server for the Web Subsystem on the Development LAN; owns the `webadm` and `webcm` UNIX accounts. To be contacted for all web server issues and any issues that require intervention of the `webadm` or `webcm` UNIX users.

System Administrator: Operates and maintains the Development LAN hardware and infrastructure software including the associated baseline inventories described in sections 2.2. and 2.3. Owns the UNIX/Linux `root` account and has UNIX/Linux superuser permissions. To be contacted for all hardware and infrastructure software changes, all UNIX/Linux user issues which cannot be addressed by the individual users or the Development LAN operator, and all general system issues on the Development LAN which are not specific to application software, database or configuration.

2.2. Generalized Hardware Description

The Development LAN hardware currently consists of a mixed network of servers and office workstations running Sun Solaris and Red Hat Linux distributions. The IDC has started a transition program from Solaris to Linux, so the Solaris servers are expected to gradually get replaced by Linux servers. One Linux machine (`oracl`) is dedicated as Development LAN database server hosting two database instances (`idcdev`, `idcdev2`). A separate cluster (`msc001dvl`) serves as central file storage cluster for all IDC LANs and NFS file systems.

The set of Linux servers is currently being extended to become the future core of the Development LAN. The servers will be grouped into machines reserved for regular data processing for each technology and development machines to be used by individual users for development, unit testing and off-line studies.

Currently the Development LAN SHI system primarily runs on Linux servers with some remaining parts still running on Solaris (eldey, flatey). The RN system primarily runs on Solaris, and the ATM system runs entirely on Linux. All software running on Linux is still compatible with Solaris, which will at least be required as long as IDC Operations is still running on Solaris machines.

As the individual machines are fully interchangeable we use symbolic machine names in this document to refer to hosts for individual data processing systems: `SHIHOST` for SHI data processing, `RNHOST` for radionuclide data processing and `ATMHOST` for atmospheric transport modeling.

All IDC LANs, including the Development LAN, have their file systems mounted from the central file storage cluster `msc001dvl` at the IDC computer center. All IDC filesystems can technically be mounted on any IDC LAN, where the top level directory is named `/${lan}`, with `$lan` being one of (dvl, tst, ops), to avoid logical conflicts with the almost identical directory structures of the other LANs. However, the Development LAN directory structure is currently mounted only on Development LAN machines.
2.3. Generalized Infrastructure Software Description

The IDC processing software is operated together with a number of third party open source or commercial software products. The most relevant examples on this list are Solaris 9 or RedHat Enterprise Server 5.1, Oracle 10g, Tuxedo 10, Netscape Enterprise Server 3.6, ClearCase 2003.06.10, Perl 5.8.8, OpenSSH 4.3.2.

The original full infrastructure software baseline for the Development LAN is available at: [http://intranet.ctbto.org/development_lan_software_checklist.pdf](http://intranet.ctbto.org/development_lan_software_checklist.pdf). The online version dates from 2003 and is not being updated. The most recent version of this document, which describes the situation before the beginning of the transition to Linux dates from July 2004 is available as [IDC-SCD]. Since then the Linux transition has brought rapid changes, and currently no up-to-date baseline exists. It is, however, desirable to create such a new baseline as soon as the Linux transition has somewhat stabilized.

2.4. Directory Structure

The IDC uses a generic directory structure for all file systems that are related to the IDC data processing software. The same directory structure is used on all IDC LANs: Operations, Testbed and Development LAN. The file systems for each LAN are identified by the name of the top level directory in this structure.

The following sections describe the high-level directory structure which is common to the three technologies SHI, RN and ATM. More detailed information on the technology-specific substructure is given in the SHI chapters of this document and in separate volumes for the RN and ATM software.

2.4.1. Top-level Directory

The top-level directory of this structure corresponds to the LAN on which the software is operated (/dvl, /tst, /ops for the Development, Testbed and Operations LANs respectively).

According to current IDC policy the /dvl file systems are mounted only the Development LAN, the /tst file systems are mounted on both the Testbed (read/write) and Development LANs (read only), and the /ops file systems are mounted on all three IDC LANs (read only on the Testbed and Development LANs).

2.4.2. Technology-Specific Directories

2.4.2.1. software, data, products, site and logs directories

The second level directories are named software, data, products and logs. These define the main directories, which contain all files used by the processing software on a given LAN.

The software directory contains the entire infrastructure and processing software including executable and configuration files for all monitoring technologies. The content of the software directory is static, i.e. it is not dynamically modified by the processing system itself but only by human operators during installation of software or configuration data changes. All technology-independent infrastructure software is located in the site subdirectory of the software directory.
The data and products directories both contain files that are dynamically written by the processing software for all monitoring technologies. Files are considered data if they are reused by the system for further processing. They are considered products if they are not further processed and/or are made available to (external) users.

The logs directory contains all log files which are dynamically generated by the IDC data processing software, except log files which are handled by the syslog server.

2.5. ClearCase Baseline, Configuration Management and Build Procedures

2.5.1. ClearCase Terminology and Role at IDC

ClearCase is the configuration management tool used for software version control on all IDC LANs. It allows users to track changes for all files which are under ClearCase control and stores the complete change history.

ClearCase VOBs define storage areas within ClearCase similar to UNIX file systems. The IDC uses a ClearCase VOB structure for its processing software which is almost identical to the high-level runtime file systems. However, the paths to the VOBs are different in ClearCase. See section 2.5.2. for a detailed description of IDC ClearCase VOBs. It is important to note that the at the IDC the runtime systems always consists of real files. The VOBs are used as a master-copy to derive a particular run-time system, but the software is never executed from the VOB.

A ClearCase version string consists of a UNIX directory and file name, a ClearCase branch name and a version number. The branch name and version number uniquely identify each version of a ClearCase element’s version tree. The version tree is the set of all historically checked-in versions of a ClearCase element (i.e. a file or a directory). Different branch names are used to identify versions for different LANs as well as versions checked in by IDC developers before integration into the processing system.

ClearCase views consist of a set of rules to "look" at version trees and select exactly one version of each ClearCase element. Specific views are configured for each IDC LAN and for individual developers and development projects.

ClearCase labels are used to tag specific file versions of interest. For example there are labels for versions which are part of IDC software patches. Other labels are used to tag those versions which are currently installed in the runtime system (i.e. outside ClearCase).

A ClearCase trigger is a procedure (typically implemented as a script) which is triggered when a specific event (typically a check-out or check-in) occurs in ClearCase. An example is a check-in trigger which automatically labels the checked-in version and installs it in the runtime directory.

The Development LAN primarily uses the devlan branch in ClearCase. Any ClearCase versions, which are created in the devlan_view, are automatically checked in on the devlan branch. The devlan_view view will automatically create a devlan branch at check-in if the branch does not yet exist in the elements version tree. In the same situation, if no devlan branch exists for an element, the devlan_view will select the latest version in the R3_tst branch that is used for the Testbed LAN. If there is no such branch for the Testbed either, the devlan_view will select the latest version on the main branch. Other
IDC/DEVLAN/OG-SHI
Page 12

IDC ClearCase views are based on a similar hierarchically organized set of selection rules. They will always check in versions on a view-specific branch and automatically create the branch if needed.

2.5.2. ClearCase VOBs
The following VOBs are relevant for operating the SHI, RN and ATM software at the IDC:

SHI Software VOBs
Maintained by the ClearCase administrator for the Testbed and Operations and by the Development LAN operators for the Development LAN:

/vobs/ibase
Repository for IDC SHI software which is not directly distributed to external users; ibase software is built in this ClearCase VOB.

/vobs/ibase_install
Installation VOB for ibase software; files are automatically installed in this VOB via the ibase build procedure, executables and UNIX man pages are manually copied from here to the runtime directories.

/vobs/states
Repository for software which was newly developed and is fully owned by the IDC; distributed to external users without any license restrictions. The states software is organized into several packages which are separately built. Installation directories are subdirectories within the same VOB, and executables are promoted from there to runtime directories as needed.

SHI scripts and config VOBs
Maintained by the Processing Engineers for the Testbed and Operations, and by the Development LAN operators for the Development LAN:

/vobs/idc/scripts
SHI scripts; executables in this VOB are not compiled; they are manually copied to the runtime scripts directory, similar to files in the config VOB.

/vobs/idc/config
SHI configuration files; to be kept synchronized with the runtime config directory; files are copied from here to the runtime directory either manually or by a ClearCase check-in trigger.

VOBs for Web Subsystem, RN and ATM Software
Maintained by the ClearCase administrator for the Testbed and Operations and by the Development LAN operators for the Development LAN:

/vobs/idc/web
Web Subsystem executables, configuration files, data and log directories; to be manually kept synchronized with the runtime web directory.
RN software repository including source code and installed executables, libraries and configuration files; files are manually copied to runtime rms directories as needed.

**VOBs to be phased out**

Maintenance responsibilities are as with *ibase*, but software in *rel/src* is built with a different procedure. Some remaining binaries for interactive SHI analysis are currently still installed from here in the runtime directory for Solaris only.

Old source code for SHI software; installed in other *rel* subdirectories (*X11*, *bin*, *doc*, *jlib*, *lib*, *scheme*, *sql*) which are separate *ClearCase* VOBs. All VOBs starting with */vobs/idc/rel* are successively replaced by *ibase* and will become fully obsolete as soon as the remaining interactive analysis applications have been ported to *ibase*.

External software needed to build IDC software on Solaris. Currently contains *libcrypto*, *libldap*, *liblber*. On IDC Linux machines these libraries are installed with the Red Hat distribution.

### 2.5.3 IDC Software Development Cycle

All IDC software changes follow a general path from developers to the Development LAN, then further to the Testbed LAN and finally to Operations. Different levels of testing and validation are involved at each level:

- **Developers:** Unit testing
- **Development LAN:** Integration testing
- **Testbed LAN:** System testing in operational environment
- **Operations LAN:** Final validation

The full IDC development cycle is described in more detail in the following twelve steps:

1. Developers develop in personal and project-specific *ClearCase* views and unit test their software.
2. Developers submit a Devlan Change Request (DCR) to the Dev LAN operators.
3. Dev LAN operators merge the changed versions from the developer branches to the devlan branch, build and install software, and perform testing in the integrated system. They report any observations and issues back to the developers.
4. Developers submit a Change Request Proposal (CRP) and Change Implementation Note (CIN) to the IDC Configuration Control Board (CCB).
5. CCB approves the change proposal and assigns the CIN for implementation on the Testbed LAN to Implementers (Configuration Manager, Processing Engineers).
6. Configuration Manager merges source code changes from the devlan branch to the testbed branch, builds the changed software applications and notifies the Processing Engineers.

7. Other Implementers implement infrastructure or database changes as assigned and notify the Processing Engineers.

8. Processing Engineers merge changes in the scripts and config VOBs to the testbed branch, install built applications, scripts and configuration files in the Testbed runtime directories and validate the changes on the Testbed LAN. They report any observations and issues back to the developers.

9. CCB approves implementation of the CIN on the Operations LAN.

10. Infrastructure changes and other preparations are implemented as assigned.

11. Processing Engineers merge changes in the scripts and config VOBs to the operations branch and install built applications, scripts and configuration files in the Operations runtime directories. Software applications are not built again for Operations such that the same executable versions are promoted as already tested on the Testbed LAN.

12. The changes are validated in Operations by the Processing Engineers and Analysts, and any observations and issues are reported back to the developers.

2.5.4. ClearCase Build and Installation Procedures

IDC software applications are built in their source code directories, and the resulting executables, library, header and man page files are installed in the ClearCase installation directories with the command `make install-idc`. For `ibase` binaries there is a separate installation VOB `/vobs/ibase_install`, while software from the states VOB is installed in `bin`, `lib`, `include` and `man` subdirectories of `/vobs/states`. The installed files are promoted with the UNIX `cp -pf` command from the ClearCase installation directories to the corresponding runtime directories. Files from the `web`, `scripts` and `config` VOBs are directly promoted to their runtime directories after check-in in ClearCase.

The general steps to merge, build and install software and configuration changes are described below:

1. **Merge changes to the devlan branch**

Before building or installing software all changes first need to be merged from a developer branch to the devlan branch. The merge procedure is the same for all ClearCase elements including source code, scripts and configuration files.

Note that directories are ClearCase elements with independent version trees and make sure that all directories are properly merged starting from the top level VOB directory.

```
ssh <user>@<any Dev LAN machine>
newgrp shicm       #webcm for /vobs/idc/web \
                   #rmscm for /vobs/idc/rms
cleartool setview devlan_view
cd <vob>/<directory>
cleartool checkout -nc <element name>
```
cleartool merge -to <element name> <version string>
cleartool checkin -nc <element name>

2. **Build ibase and states software with autotools**

It is possible to build either entire packages or just specific libraries and applications by changing to the relevant directory of interest. The `make` commands will run recursively for all subdirectories. *ibase* is set up as a single *autoconf* package, while the *states* software is organized in several packages, which need to be built separately. If *autotools* files or `Makefile.am` files have been modified first follow the procedure described in section 2.5.5. to re-create the platform-specific `Makefiles`.

```bash
ssh <user>@<Linux or Solaris Dev LAN machine>
    # make sure to build
    # on the desired target platform
newgrp shicm
cleartool setview <platform-specific build view>
    # devlan_view,
    # devlan_solaris_view,
    # dls_build_64_view, dls_build_32_view
cd <vob>/<directory>  # ibase: /vobs/ibase
make clean    # provided that configure was already run earlier
make
make install-idc
```

For old software in `/vobs/idc/rel/src` use the `devlan_view`, make sure to build on Solaris only, mount the `admin` VOB (cleartool mount /vobs/idc/admin) and run the commands `make clean -clearcase; build.pl -v -ab` instead of the regular `make` commands.

3. **Install files in runtime directories**

Built software executables, UNIX man pages, shared libraries, scripts and configuration files need to be promoted to the runtime system. Make sure to first shut down any running instances of applications on the target platform before installing changed files. Restart the applications after the file installation if needed.

```bash
ssh <cmss>@<any Dev LAN machine>  # webcm for web subsystem,  # rmscm for rn software
cleartool setview <platform-specific build view>
    # devlan_view,
    # devlan_solaris_view,
    # dls_build_64_view, dls_build_32_view
cd <runtime directory>  # ibase: /${lan}/software/shi/rel/bin/<platform-specific subdirectory>
diff <vob>/<directory>/<filename> <filename>    # should differ
cp -pf <vob>/<directory>/<filename> <filename>
diff <vob>/<directory>/<filename> <filename>    # should not differ
```
Compiled software is only installed from ClearCase installation VOBs and directories, never from source code VOBs or directories. Web, script and config files are directly installed after check-in, without compilation.

Current platform-specific runtime directories for executables built in the ibase and states VOBs are:

$(lan)/software/shi/rel/bin/Linux-x86_64  #Linux 64-bit
$(lan)/software/shi/rel/bin/Linux-x86_64_32 #Linux 32-bit
$(lan)/software/shi/rel/bin/SunOS-sparc  #Solaris

2.5.4.1. VOB Specifics

states VOB:
There is a package-specific labelling script to label installed builds for the cdtools, cd2w and gbse packages. Run the script after make install-idc:

/vobs/idc/dev/SHI/scripts/pl/label_cdtools.pl

Old rel/src VOB:
Verify that the make is executed as /usr/ccs/bin/make. Add /usr/ccs/bin before /opt/OSS/bin in your PATH environment variable if needed.

2.5.5. More on autotools

GNU autotools are a set of utilities (aclocal, autoheader, autoconf, automake) which help to develop and build cross-platform software on UNIX-like operating systems. At the IDC they are used to build software in the ibase and states VOBs for Linux 64-bit, Linux 32-bit and Solaris platforms. While the ibase VOB consists of only one software package the states VOB consist of individual packages for various sets of applications.

Figure 2 gives an overview over the procedure to update all files which are needed to build a software package on a particular platform. The following factors need to be considered for particular builds:

- The platform and operating system.
- The ClearCase view.
- Was the file configure.ac modified?
- Have one or more Makefile.am files been modified?
The following steps describe the procedure to create or update all autotools files for a particular software package, platform and ClearCase build view.

1. Login on a machine with the desired platform and set the proper ClearCase build view:

   ssh <user>@<Linux or Solaris Dev LAN machine>
   # make sure to build on the
   # desired target platform

   newgrp shicm
cleartool setview <platform-specific build view>
   # devlan_view,
   # devlan_solaris_view,
   # dls_build_64_view, dls_build_32_view

   cd <vob>/<top-level package directory> # ibase: /vobs/ibase

2. If configure.ac has been modified update the autotools configure script:

   aclocal; autoheader; autoconf

3. If one or more Makefile.am files have been modified update the Makefile.in files:

   automake  # automake --foreign in the states VOB
   # automake <vob>/<directory>/Makefile
   # to update only one specific Makefile.in

---

**Figure 2 autotools build procedure**

Platform depend flags can be derived from /home/misc/cmss/build_ibe
4. Execute the `configure` script with the following options to create platform-specific Makefiles:

```bash
configure --prefix=/vobs/<install directory> --enable-shared=no \\   --enable-idc
```

4.a To direct the compiler to build 32-bit executables on 64-bit systems:
Add the option `--enable-32bit` to the `configure` command line.

4.b To link interactive SHI analysis applications in `ibase` to `libipcnt` instead of `libipc`:
Linking to `libipcnt` will generate executables for IPC communication via the `nontux` framework, while linking to `libipc` will generate executables for IPC communication via `Tuxedo`.

Add the option `--with-nontux` to the `configure` command line.

4.c For packages in the `states` VOB:
Add the following options to the `configure` command line:

```bash
CFLAGS="-g -DENVIRONMENT" CPPFLAGS="-I/vobs/states/include \\
   -I/vobs/states/include/CDtools" LDFLAGS=-L/vobs/states/lib
```

4.d On Solaris use the following `configure` command line:

```bash
configure --prefix=/vobs/<install directory> --enable-shared=no \\   --enable-oracle9 CC=/opt/SUNWspro/bin/cc SITE_HOME=/opt/OSS \\
   F77=/opt/SUNWspro/bin/f77 \\
   LDFLAGS="-L/opt/SUNWspro/lib -L/opt/OSS/lib"
```

4.e For Solaris packages in the `states` VOB:
Add the following options to the `configure` command line for Solaris:

```bash
CFLAGS="-g -DENVIRONMENT" CPPFLAGS="-I/opt/OSS/include \\
   -I/vobs/states/include -I/vobs/states/include/CDtools" \\
   LDFLAGS="-L/opt/OSS/lib -L/vobs/states/lib" LIBS=-ldl
```

Once the Makefiles have been updated `make clean; make; make install-idc` can be run as described in section 2.5.4. to build single libraries or applications or entire software packages.
2.6. Change Request and Problem Reporting Procedures

2.6.1. Introduction

Software and configuration changes on the Development LAN are typically requested by software developers or other Development LAN users. Such changes are requested by sending a Development LAN change request (DCR) containing the specific set of ClearCase versions to be installed (see procedure in section 2.6.2.).

The Development LAN hardware, NFS file systems and the infrastructure software are maintained by the system administrators. Any changes to the infrastructure baseline will typically need a system-wide integration test to verify that no applications and processing subsystems are negatively affected by the change. All such changes need to be coordinated with the system administrators, the Development LAN operators and the developers of any software that may be affected by the change. See section 2.6.3. for requesting and implementing baseline inventory changes on the Development LAN.

Changes to Development LAN procedures and problem reporting procedures are described in sections 2.6.4. and 2.6.5.

2.6.2. Requesting and Implementing Changes on the Development LAN

The Development LAN serves as an integration and test platform for IDC software. Developers and testers of new software frequently request changes to the application software or configuration to be installed on the Development LAN.

- To request a software or configuration change send a Development LAN Change Request (DCR):
  
  ssh <user>@<devlan_machine>
  $(SCRIPTSBIN)/devlan-request.pl

- All fields of the form in the devlan-request window need to be filled-in. Provide all relevant information including a complete list of specific files and ClearCase versions to be merged and any other instructions needed to correctly implement the change. After editing all input fields press the Submit button to send the DCR.

- The new change request will get an identifier assigned (DCR-<number>) and will be available for tracking at http://eldey.idc.ctbto.org:5678/report/all_dcrs.html. The original DCR will automatically be sent as an email message to both the requestor and the Development LAN operators.

- The Development LAN operators will implement the change request depending on current priorities and other integration and test activities on the Development LAN. They will contact the requestor to clarify any implementation questions if needed. The Development LAN operators will update the status of the DCR when it is implemented. The status update will include any specific observations made when implementing the change on the Development LAN. It will automatically be sent as an email notification to the requestor and the Development LAN operators. The utility for the Development LAN operator to update and respond to DCRs is a Perl script named devlan-respond.pl.
• In order to add additional information or modify the existing DCR the requestor can either contact the Development LAN operators directly, send the information per email or use devlan-respond.pl to update the DCR status to MODIFIED.

• Once the change is fully implemented on the Development LAN, the requestor should monitor the processing results and/or start the planned software test.

• Experience gained from the implementation on the Development LAN should be included in the CIN to later implement the change on the Testbed and Operations LANs.

### 2.6.3. Changes to the Baseline Inventory

Baseline changes are implemented by the responsible person who maintains the specific baseline as follows:

• The system administrators maintain the hardware and infrastructure software baselines for all IDC LANs. They are also responsible for maintaining the NFS file systems and the general system environment as far as UNIX/Linux root access is needed for such maintenance.

• The ClearCase administrator maintains the ClearCase VOBs, views, labels, branches and access permissions.

• The software developers are responsible for maintaining the application software baseline and the application-specific configuration.

• The Development LAN operators maintain the application software configuration and all scripts, software utilities, cron jobs, configuration files, data files and database tables supporting the integrated data processing system on the Development LAN. They also maintain the directory structure, the database schema and the generic system environment as far as not within the responsibility of the system or database administrators.

• To request a baseline change contact the person who is responsible for the specific baseline. To contact the Development LAN operators send a Development LAN change request. Determine together with the responsible implementer which other system changes may be needed as a consequence of the baseline change.

• Depending on the type of change and its impact on the system the requestor and the responsible implementer may have to get management approval for implementing the baseline change. Contact the Head of the Software Integration unit to determine the formal steps that apply to the specific case.

• If the change is approved inform all responsible people who will have to contribute to the implementation and validation of the baseline change and of any related changes that have been determined to be necessary.

• Inform the Development LAN operators of the changes to be implemented and of the testing requirements after implementation of the changes. The Development LAN operators will coordinate the schedule to implement and test the changes.

• When the implementation is complete the implementers verify the correct implementation of their changes. The Development LAN operators restart the data processing system and verify that all subsystems successfully process data. More specific tests need to be planned in advance.
• The person maintaining the baseline that has been changed updates the corresponding baseline document.

• After implementation, testing and documentation are complete on the Development LAN submit a CIN to request the same change on the Testbed and Operations LANs. Include all experience gained from the Development LAN implementation to refine the implementation procedure for the Testbed and Operations LANs.

2.6.4. Changes to Development LAN Procedures
Changes to Development LAN procedures may be useful to improve the usability of the procedure and to reduce any ambiguities that may exist. In case it is determined that a procedure is not applicable it should be changed in order to reflect the current status.

• Depending on the type and impact of the requested change, approval from the Head of the Software Integration unit may be needed.

• Replace the procedure by the improved version in the Development LAN operator's Guide or ask the maintainer of the document to do so.

• When the changed procedure is applied, verify that the expected improvement is achieved. If this is not the case change the procedure again as needed.

2.6.5. Problem Reporting
The IDC uses ClearQuest to document and track software problems. To report a new problem:

• Check in ClearQuest if the problem already has been reported before. Existing problem reports should not be duplicated, but new information should be attached to the problem report in ClearQuest. Contact the ClearQuest administrator to get assistance with attaching information to existing problem reports.

• Collect all relevant information (and avoid any non-relevant information) that is necessary to reproduce the problem offline (log file sections, database entries, information on the specific scenario, user actions or environment to reproduce the problem, etc.).

• If there is no existing problem report in ClearQuest send an email to the ClearQuest administrator describing the nature of the problem and specifying the application for which the problem occurred. Attach or refer to all collected information.

• Refer to the IDC ClearQuest procedures and development cycle diagrams for information on the next stages of addressing and resolving problem reports. Generally, the problem report will be assessed and categorized, and depending on the type of problem the responsible manager will assign a developer in order to resolve the problem. Once a solution is available and has been unit-tested by the developer it will first be installed and tested on the Development LAN before being promoted to the Testbed and Operations LANs. The software request lifecycle diagram is available at: http://idc030.idc.ctbto.org:8000/GIF/SOFTWAREREQUEST-LIFECYCLE-LATEST.jpg
3. SHI DATA ACQUISITION AND PROCESSING

3.1. Introduction

SHI stands for Seismic, Hydroacoustic and Infrasound, the three waveform technologies used by the CTBTO to monitor the Comprehensive Test Ban Treaty. These three technologies have in common, that the respective types of IMS stations, namely

- **S**: seismic arrays and three-component seismograph stations to record seismic waves travelling through the solid earth,
- **H**: hydrophone arrays to record sound waves travelling along a waveguide channel in the oceans, and
- **I**: infrasound stations to record infrasonic vibrations of the Earth’s atmosphere,

acquire digitally sampled time-series data and transmit them continuously in real time (primary S, H and I stations) or as time segments upon automatically generated requests (auxiliary S stations) to the IDC.

At the IDC the SHI software receives continuous and segmented data and automatically processes all data to detect seismic, hydroacoustic and infrasound arrivals at the IMS stations and determine location, time and magnitudes of the events from which the signals originated. It further generates a range of derived data products and graphical plots, archives all data and data products and makes products available to external authorized users via email messages, subscriptions and the IDC secure website. The SHI software also includes a suite of interactive analysis software tools, which are used by human analysts to review and where necessary to correct and improve the automatic processing results.

On the Development LAN data is typically received from other IDC LANs and only automatic processing is routinely run. The interactive processing software is available for testing on demand.

3.2. Baseline Inventory

3.2.1. Hardware Baseline

The automatic SHI processing software can be operated either on a single machine or on multiple machines which may be distributed across a heterogeneous network including different hardware architectures and operating systems. Automatic processing is controlled by the Distributed Application Control Subsystem (DACS) which consists of a set of in-house maintained applications that depend on the commercial middleware software *Tuxedo*. *Tuxedo* uses a central configuration file, the so-called *ubb* file, which defines all DACS-controlled processes, organizes these processes into groups and assigns groups to symbolic hosts to run on. All DACS processes defined in the *ubb* file can be started, stopped and monitored both from the *tmadmin* command line interface to *Tuxedo* and from the IDC-maintained GUI tool *tuxpad*. Thus, if the *ubb* file is configured accordingly, automatic SHI processes can easily be moved with *tuxpad* between individual machines and platforms including, for example, Linux and Solaris machines.

On the Testbed and Operations LANs the *ubb* file is only rarely changed and automatic SHI processing is typically run in a distributed mode across five dedicated machines.

---

May-08
On the Development LAN a second down-sized version of the automatic SHI system named SEL0 system is running on separate machines. More on the SEL0 system can be found in section 3.2.7.3 on network processing.

Interactive SHI processing consists of a suite of interactive analysis GUI tools, which are run by individual analysts. This is typically done locally on the analyst’s workstations. Interactive analysis is performed routinely on the Operations LAN, while interactive analysis on the Testbed and Development LANs is performed only on demand, typically for testing purposes or for dedicated studies.

The SHI processing system uses two database instances, one operational and one archive database instance. On the Development LAN both database instances are hosted on a single database server, while the Testbed and Operations LANs use two separate database servers on separate machines to host their operational and archive database instances. The names of the database instances on the three IDC LANs are as follows:

<table>
<thead>
<tr>
<th>IDC LANs</th>
<th>Operational Database</th>
<th>Archive Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development</td>
<td>idcdev</td>
<td>idcdev2</td>
</tr>
<tr>
<td>Testbed</td>
<td>fogo</td>
<td>fako</td>
</tr>
<tr>
<td>Operations</td>
<td>mauie</td>
<td>moorea</td>
</tr>
</tbody>
</table>

3.2.2. Application Software Baseline

The automatic SHI processing software is run by the three UNIX/Linux users: the three users auto, cmss and webcm are maintained by the Development LAN operator. Utilization of the webcm user for the Web Subsystem is a feature specific to the Dev LAN – in other IDC LANs the webadm user is responsible for this task. The web server for the Web Subsystem is run by the UNIX user webadm and maintained by the Web Administrator.

The cmss user runs the CDTools for continuous waveform data acquisition and forwarding, the Archive Subsystem and a number of cron jobs to migrate data to the archive database.

auto user runs the automatic and interactive Distributed Application Control Subsystem (DACS), all automatic processes which are controlled by the DACS, the Message, Retrieve and Subscription Subsystems and various cron jobs.

The interactive SHI analysis software is run by individual analysts from their UNIX accounts, and by other users who wish to interactively analyze SHI data.

For the most SHI software applications or subsystems there are corresponding software design and user manuals available in the existing IDC documentation. All Software User Manuals are labeled [IDC6.5.x], and Software Design Documents are labeled [IDC7.x.y]. The current versions of all IDC software documents are electronically available at the IDC products and services web site:

http://kuredu.ops.ctbto.org
3.2.3. Software Location

All SHI software applications are located below the directory ${lan}/software/shi/rel ($RELDIR). Executable files are installed in the bin subdirectory, which contains another set of subdirectories for platform-specific executables: Linux-x86_64 for 64-bit Linux binaries, Linux-x86_64_32 for 32-bit Linux binaries, SunOS-sparc for Solaris 32-bit binaries. The source code files are available only under ClearCase in the ibase and states VOBs – they do not exist in the run-time file system. The SHI software applications are grouped into several subsystems for automatic processing, interactive processing, process control, system monitoring, etc. These subsystems are explained in section 3.2.7, and more detailed information on the directory structure is provided in section 3.2.5.

All SHI scripts, which are part of the application baseline are located in the directory ${lan}/software/shi/scripts/bin (= $SCRIPTSBIN). The scripting languages used for scripts are sh, csh, Perl and Tcl/Tk.

UNIX/Linux manual pages are available for most applications and scripts of the SHI software. They are located in rel/doc subdirectories and can be accessed with the UNIX/Linux man command:
man <application or script name>

3.2.4. Software Configuration Baseline

The high-level parameter files are located in the system_specs directory and define parameters that are shared by multiple applications within a subsystem. These high-level parameter files are automatic.par, DFX.par, dacs.par, interactive.par, miscspecs.par, msgs.par as well as a number of other files.

Further, all individual parameter files include the top-level parameter file shared.par via the IMSPAR environment variable (see further below). Ideally, shared.par should be the only parameter file in the entire configuration structure that contains explicit path names to system directories. All other parameter files should use these high-level parameters to reference symbolic directory paths.

Similarly, database connection strings for all database accounts are defined in parameter files for each CMS_MODE (see section 3.2.6.) All other parameter files use the symbolic database account names defined in analysis.par, process.par or public.par, which reside in the system_specs directory. The environment variable IMSPAR points to the appropriate file depending on CMS_MODE. Individual parameter files include the proper database parameter file by a reference to IMSPAR, and the database parameter files include in turn shared.par.

The following two lines in individual parameter files include all relevant high-level configuration files:

par=$(IMSPAR) #points to analysis.par, process.par or public.par,\ #which inherit variables defined in shared.par
par=$(<symbolic_parfile_name>)
#symbolic names of high-level
#parameter files are defined in shared.par
#for example par=$(AUTOMATIC)
Maintaining the hierarchy of configuration files is crucial for the long term maintainability of the SHI data processing system.

When new software is implemented on the Development LAN it always needs to be integrated with this existing configuration structure in order to keep the system coherent during repeated software development cycles.

See [IDC6.2.4] for a description of IDC configuration data files.

### 3.2.5. Directory Structure

The general top-level directory structure for the Development LAN is described in chapter 2. The SHI-specific top level directories follow this generic structure and are listed in the table below. For an overview of the complete Development LAN directory structure see section 2.4. Directory levels are counted starting from the top level (level 1), which is $(lan)=/dvl for the Development LAN.

<table>
<thead>
<tr>
<th>Top Level (Level 3) SHI Directories</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(lan)/data/shi = $(SHI_DATADIR)</td>
</tr>
<tr>
<td>data directories and files used by automatic SHI processing software</td>
</tr>
<tr>
<td>$(lan)/products/shi = $(SHI_PRODUCTDIR)</td>
</tr>
<tr>
<td>SHI data products</td>
</tr>
<tr>
<td>$(lan)/logs/shi = $(LOGDIR)</td>
</tr>
<tr>
<td>SHI log file area</td>
</tr>
<tr>
<td>$(lan)/software/shi</td>
</tr>
<tr>
<td>installed SHI software (binaries, libraries, scripts, configuration files)</td>
</tr>
<tr>
<td>$(lan)/software/shared/web</td>
</tr>
<tr>
<td>Web Subsystem software; shared by SHI, RN and ATM software</td>
</tr>
<tr>
<td>$(lan)/data/shared/messages = $(MSGDIR)</td>
</tr>
<tr>
<td>data, request and subscription messages; shared by SHI, RN and ATM software</td>
</tr>
</tbody>
</table>

#### 3.2.5.1. software Directory

The SHI software is structured into a ./rel directory containing source code and built files for all SHI applications, a ./config directory containing the entire SHI configuration structure and a ./scripts directory containing released scripts.
### 3.2.5.2. rel Directory

<table>
<thead>
<tr>
<th>Level 5: SHI software release directory structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(lan)/software/shi/rel/bin</td>
</tr>
<tr>
<td>./bin = $(RELBIN)</td>
</tr>
<tr>
<td>./SunOS-sparc</td>
</tr>
<tr>
<td>./Linux-x86_64</td>
</tr>
<tr>
<td>./Linux-x86_64_32</td>
</tr>
<tr>
<td>./doc</td>
</tr>
<tr>
<td>./lib</td>
</tr>
<tr>
<td>./scheme</td>
</tr>
<tr>
<td>./sql</td>
</tr>
<tr>
<td>./jlib</td>
</tr>
<tr>
<td>./X11</td>
</tr>
</tbody>
</table>

The $(lan)/software/shi/rel/bin directory contains symbolic links of all executable files to a wrapper script called isaexec.sh. This script was designed to allow usage of the same directory tree within a heterogeneous network. It determines the architecture of the calling environment and automatically executes a proper binary for each platform (Linux, Solaris) and architecture (Sparc, i686, x86_64, etc.). The binaries reside in architecture-specific subdirectories of the bin directory (SunOS-sparc, Linux-x86_64, etc.). The isaexec mechanism guarantees that the proper binaries will be automatically selected on each machine in an inhomogeneous network sharing a common file system and system environment. The user can always refer to binaries in the bin directory, regardless of the machine or a particular software platform. The $PATH environment variable needs to contain only the $(lan)/software/shi/rel/bin location in order to allow access to all SHI data processing applications.

Example: The actual mig_bull executables reside in platform specific subdirectories:

```bash
$(lan)/software/shi/rel/bin/SunOS-sparc/mig_bull
$(lan)/software/shi/rel/bin/Linux-x86_64/mig_bull
$(lan)/software/shi/rel/bin/Linux-x86_64_32/mig_bull
```

The $(lan)/software/shi/rel/bin/mig_bull is a symbolic link to isaexec.sh. The isaexec script determines the platform and executes ./Linux-x86_64/mig_bull on 64-bit Linux machines, ./Linux-x86_64_32/mig_bull on 32-bit Linux machines and ./SunOS-sparc/mig_bull on SunOS on Solaris machines. The directory $(lan)/software/shi/rel/bin contains a link to the isaexec script:

```
lrwxrwxrwx 1 cmss  cmss  10 Jun 8 15:20 mig_bull -> isaexec.sh*
```

### 3.2.5.3. config Directory

All configuration files for the SHI processing system are located below the directory $(lan)/software/shi/config that corresponds to the $CMS_CONFIG variable in the software parameter files. The config directory has subdirectories for application configuration files (app_config), geophysical data files describing earth specifications.
(earth_specs), host specific configuration files (host_config), station specific configuration files (station_specs) and system specific configuration files (system_specs).

The table below shows the top level configuration subdirectories. They group major types of configuration files and contain further levels of subdirectories, e.g. for groups of applications, etc.

<table>
<thead>
<tr>
<th>Level 5: SHI configuration directory structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(lan)/software/shi/config = $(CMS_CONFIG)</td>
</tr>
<tr>
<td>./app_config</td>
</tr>
<tr>
<td>./earth_specs</td>
</tr>
<tr>
<td>./host_config</td>
</tr>
<tr>
<td>./station_specs</td>
</tr>
<tr>
<td>./system_specs</td>
</tr>
</tbody>
</table>

The configuration files in the app_config, earth_specs, host_config and station_specs directories all include so-called high-level parameter files by reference. This is done in order to avoid duplication of information as much as possible and to simplify the implementation of configuration changes.

3.2.5.4. data and products Directories

The next table lists the SHI data directories in $(lan)/data/shi. There are separate data directories for auxiliary station waveforms, beams, continuous waveform data, segment archive data and the threshold monitoring session data.

<table>
<thead>
<tr>
<th>Level 4: SHI data directories</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(lan)/data/shi = $(SHI_DATADIR)</td>
</tr>
<tr>
<td>./aux</td>
</tr>
<tr>
<td>./beams</td>
</tr>
<tr>
<td>./cds</td>
</tr>
<tr>
<td>./segment</td>
</tr>
<tr>
<td>./tm_session</td>
</tr>
</tbody>
</table>

SHI product directories are QTrace, reports and tm for QTrace plots, squal and Xlogger reports and threshold monitoring products (PostScript plots). They are subdirectories of $(lan)/products/shi as listed below.

<table>
<thead>
<tr>
<th>Level 4: SHI products directories</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(lan)/products/shi = $(SHI_PRODUCTDIR)</td>
</tr>
<tr>
<td>./QTrace</td>
</tr>
<tr>
<td>./reports</td>
</tr>
<tr>
<td>./tm</td>
</tr>
</tbody>
</table>
3.2.5.5. shared Directory

There are two major directories which are shared between the SHI and the RN processing system. These are the directories for the Web Subsystem and for messages, which are received or sent by the Message Subsystem. They are subdirectories of $(lan)/software/shared and $(lan)/data/shared/messages respectively.

<table>
<thead>
<tr>
<th>Level 4: shared directories used by SHI software</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(lan)/software/shared/web</td>
</tr>
<tr>
<td>$(lan)/data/shared/messages = $(MSGDIR)</td>
</tr>
</tbody>
</table>

3.2.6. SHI User Environment

There is a standard SHI user environment based on csh, which should be used by all users who want to develop, test or use any SHI software on the Development LAN. The standard user environment is created in the user's home directory by the script mkCMSuser. The user needs to choose one of the available modes: process, analysis or public. The mode will be stored in the CMS_MODE environment variable, which points to the appropriate parameter file defining database access strings for various accounts depending on the mode. The following modes are available:

**process:** This mode is appropriate for users who run the automatic SHI processing software on the Development LAN (i.e. cmss and auto). For developers this mode is only recommended for temporary use during unit tests that are coordinated with the Development LAN operator, because SHI applications will write to the same database tables as the continuously operated automatic processing software.

**analysis:** This mode is appropriate for users who run interactive SHI analysis applications.

**public:** This mode is appropriate for all other users who need database insert/update permissions only for the public database accounts (i.e centre@idcdev or centre@idcdev2).

In order to initially set the proper environment log into a Development LAN machine and run the mkCMSuser script with the required argument mode=<mode> to set up the Development LAN user environment:

```bash
ssh <user>@<devlan_machine>
setenv CMS_CONFIG $(lan)/software/shi/config
$(lan)/software/shi/scripts/bin/mkCMSuser mode=<mode>
exit
```

The mkCMSuser script copies default environment files including the generic.cshrc file from $(CMS_CONFIG)/system_specs/env to the user's home directory. A new .cshrc file in the user home directory will set the CMS_CONFIG and CMS_MODE environment variables and source the standard system environment from $(CMS_CONFIG)/system_specs/env/global.env.
The `global.env` file defines the remaining Development LAN environment variables including IMSPAR, which points to the parameter file defining database connection strings depending on CMS_MODE. Finally these database-specific parameter files include the top-level system configuration file `shared.par`. As it can be easily seen, this will work only in case the user has the default login shell set to `/bin/csh`.

To switch to the environment to a different CMS_MODE, it is first necessary to set the new CMS_MODE and then source the `.cshrc` file in the home directory, which will set the remaining environment as described above. For example, to switch from the process environment to the analysis environment:

```
ssh <user>@<devlan_machine>
setenv CMS_MODE analysis
source ~/.cshrc
```

To change the system environment, it is required to change the appropriate environment variable in the `global.env` file. However, to apply the environment change also to `cron` and Tuxedo a few additional steps are needed. The local Tuxedo configuration on every interactive and automatic processing host also uses local environment files, which need to be copied from the global environment files when those are changed. Tuxedo as well as `cron` use a Bourne shell environment, thus the `global.env` file needs to be translated to a corresponding `global.shenv` file. This is done with the script `csh2sh`:

```
setenv CMS_MODE process # or analysis for interactive processing \\
source global.env # to source the automatic or interactive \\
# system environment
csh2sh global.env > global.shenv # translates csh to sh variables
```

To keep global environments and the Tuxedo configuration for automatic processing synchronized the following steps have to be performed:

- Login as user auto on each DACS machine and copy `global.shenv` to `tux.env`.

```
  cd /var/tuxedo/IDC3_process/config
  cp -pf $(lan)/software/shi/config/system_specs/env/global.shenv \
  tux.env
```

- Restart all automatic DACS applications, as described in section 3.3.4.

The Tuxedo hosts for interactive processing need to be configured for the analysis environment. Set the auto environment for analysis, then resolve the `global.env` file again and copy to the local configuration directory on each interactive Tuxedo host:

```
ssh auto@<devlan_host>
setenv CMS_MODE analysis
source ~/.cshrc
imspar analysis-hosts # lists all interactive Tuxedo hosts
scp -pf `csh2sh $(CMS_CONFIG)/system_specs/env/global.env` \
<interactive_tux_host>:/var/tuxedo/IDC3_analysis/config/tux.env
```

Restart the interactive DACS processes on each host after changing `tux.env`, as described in section 3.3.5.
Generally, restart all continuously running SHI processes (see section 3.3.1. including the automatic and interactive DACS, Message and Subscription Subsystems and CDTools after any change to the system environment.

The command `imspar` can be used to resolve symbolic names for database accounts and all parameters, which are defined in `shared.par`, e.g.:

```plaintext
imspar PUBLICDB    #returns: centre/data@idcdev
imspar LOGDIR      #returns: $(lan)/logs/shi
imspar            #without argument returns the current
                   #base parameter file, e.g. process.par
```

The command `sqlwrap` can be used in combination with the symbolic database account names to log into the corresponding account on the Development LAN database. For example, to log into the `centre` account at `idcdev` or at `idcdev2`:

```plaintext
sqlwrap PUBLICDB    #log into centre@idcdev
sqlwrap PUBLIC-ARCHDB #log into centre@idcdev2
```

### 3.2.6.1. Automatic Processing User Environment

For automatic processing users the `CMS_MODE` variable has to be set to `process`. After that the user will be able to execute various components of the automatic processing software and have access to the relevant processes. Also the relevant database accounts become accessible for that user. For processes to execute successfully the user must belong to the relevant UNIX groups `‘auto’` and `‘cmss’`.

### 3.2.6.2. Interactive Processing User Environment

To set the environment for an interactive user, i.e. an SHI analyst or a user performing this role it is necessary to run `mkCMSuser` with the `mode=analysis` parameter. Also the user must belong to the `interactive` group. To configure and start Tuxedo for interactive processing the following command must be executed on the local machine as user `‘auto’`:

```
crInteractive mode=analysis task=rebuild restart
```

### 3.2.7. Data Acquisition and Processing

#### 3.2.7.1. Overview of SHI Data Processing

The SHI software receives continuous and segmented waveform data and automatically processes all valid incoming data. Figure 3 symbolically shows all subsystems that form part of SHI data processing. For most of these subsystems, Software Design Descriptions (SDDs) and Software User Manuals (SUMs) exist as part of the IDC documentation. These are referenced in the following as appropriate. See the References section for a complete list.

Continuous data is continuously received from seismic, hydroacoustic and infrasound primary stations, stored locally and registered in the database. This is accomplished by the Continuous Data Acquisition Subsystem (CDTools, see [IDC-CDT-SUG]).
The data are then processed in regular time intervals in order to detect seismic, hydroacoustic and infrasound arrivals and determine location, time and magnitudes of the events from which the signals originated.

In a first processing step called Station Processing, time-series data from each individual station are processed to derive and characterize arrivals (see section 3.2.7.2.). Then, in a second processing step called Network Processing, results from Station Processing are associated to groups of arrivals to derive, locate and characterize event origins (see section 3.2.7.3.).

On all IDC LANs there are three (SEL1, SEL2 and SEL3) iterations of the network processing. On the Development LAN there are two additional, separate network processing pipelines - SEL0 and SEL01 (see section 3.2.7.3. for details). After each network processing iteration, based on the processing results additional data is requested from the seismic auxiliary stations by the Retrieve Subsystem (see section 3.2.7.4.). When the requested data have been received and parsed by the Message Subsystem (see [IDC7.4.2],
[IDC6.5.19]), they will be taken into account in the next iteration of Network Processing, so that event locations become successively more refined.

The last iteration of network processing, SEL3, typically processes data intervals twelve hours after real time. After SEL3 processing has completed, interactive analysis can start. This step is not routinely performed on the Development LAN, but it may be performed on demand, while raw waveform data are still kept on-line. (Purging of waveform data occurs 10 days after real time on the Development LAN.) As an alternative to analyst review, interactive processing may be simulated by automatically copying the results of automated processing from the SEL3 into the LEB database account (see section 3.2.7.5. Post-Analysis processing is another automated processing step which follows after analyst review, or more typically on the Development LAN, after automated LEB simulation.

The Web Subsystem publishes processing results on the IDC’s data and product websites. Another way for data users to receive IDC data and products is to submit requests via Email. Such request messages are handled by the Message Subsystem, while the Subscription Subsystem (see [IDC7.4.4], [IDC6.5.21] fulfills standing orders for data and products.

In order to avoid overloading the operational database, aging table rows are migrated to an archive database by Data Migration. After a configurable purge delay, table rows that have been successfully migrated will be removed from the operational database by Data Purging. On the Operations LAN an Archiving Subsystem (see [IDC7.5.1], [6.5.22] eventually writes old data to a mass storage device. There is no requirement to permanently archive data from the Development LAN.

Some of the subsystems in SHI data processing are described in more detail below.

3.2.7.2. Station Processing

Automatic station processing (see [IDC5.2.1] chapters 2, 3 and 4) runs on a regular basis on the Development LAN. Components of Station Processing are symbolically shown in Figure 4. DFX (see [IDC7.1.1] and StaPro (see [IDC7.1.12], [IDC6.5.11]) are the signal processing applications that detect and characterize arrivals in the waveform data. The rest of the components belong to the Distributed Application Control System (DACS, see [IDC7.3.1], [IDC6.5.2]). Using configuration data files only, the components are configured to operate in a pre-defined sequence (a so-called pipeline) on selected chunks of time-series data (so-called intervals). A pipeline includes error handling and retries and may comprise branching, forking and joining, however, the Station Processing pipeline is relatively simple:

- **tis_server**, a data monitor, continuously monitors incoming waveform data. When enough fresh data have arrived, it creates a new interval in the IDCX.interval table using db_server.
- **interval_router** enqueues a message in the DFX Tuxedo queue.
- **TMQFORWARD**, a forwarding agent that is part of the Tuxedo distribution, dequeues the message from the queue and requests the DFX service from the application manager tuxshell.
- **tuxshell** then calls the DFX application with proper parameters and waits until its data interval has been processed.
If DFX processing has been successful, `tuxshell` updates the status of the pertinent row in the `interval` table and enqueues a message to the `StaPro Tuxedo` queue.

Another `TMQFORWARD` picks it up, another `tuxshell` instance launches the `StaPro` binary. As soon as `StaPro` completes with a favourable return code, the station processing for that particular interval is over.

**Figure 4 Components of station processing**

### 3.2.7.3. Network Processing

Network processing (see [IDC5.2.1] chapter 5) pipelines are constructed using components from among the groups shown in Figure 5.

- Global Association (`GA`, see [IDC7.1.4], [IDC6.5.12]) is a family of applications that derive event locations from a global set of arrivals, through the steps of selecting eligible arrivals, forming, rejecting and accepting association hypotheses, and inverting arrivals using travel time curves to derive locations and resolving conflicting hypotheses.

- The Retrieve Subsystem (see [IDC7.4.3], [IDC6.5.20]) serves to request data from auxiliary seismic stations based on predicted arrivals at these stations, where the predictions are made on the basis of preliminary locations. It is discussed in more detail in the next section.
In the last iteration of network processing, a DFX instance under the name of Beamer (see [IDC7.1.1, chapter 4, section 2]) is run to prepare optimized beams for seismic array stations to support Analyst review.

Finally, the Threshold Monitoring Subsystem (see [IDC7.1.14], [IDC6.5.14]) implements a technique to calculate the (dynamically fluctuating) detection threshold for seismic events for every point on the globe and every point in time.

In addition to the standard network processing pipelines on the Development LAN there are two separately running versions of the automatic network processing pipelines, SEL0 and SEL01:

The SEL0 system is used to test early bulletin production as soon as the continuous IMS data is received. The SEL0 event bulletin is produced with delays of around 20-25 minutes relative to real-time. Only primary seismic data is used in SEL0 processing, as hydro and infra
stations are unable to contribute arrivals in time, due to longer delays in propagation from the source.

The SEL01 network processing pipeline was added to run in parallel to the SEL0 network processing pipeline to investigate the effect on the bulletin of treating auxiliary stations as primary stations in network processing.

The so-called SEL0 system runs continuously and consists of seismic and hydro-acoustic station processing pipelines, plus the two (SEL0, SEL01) network processing pipelines described above. The SEL0 and SEL01 bulletins are available on the IDC Intranet on an automatically updated web page:

http://eldey.idc.ctbto.org:8888/Events/index-sel0.html
http://eldey.idc.ctbto.org:8888/Events/index-sel01.html

Neither SEL0 nor SEL01 interact with the normal SEL1 processing. Independent directories for configuration data files and for the Tuxedo environment are used to prevent such interaction:

$(lan)/software/shi/idc0/config
/var/tuxedo/SEL0_process

3.2.7.4. Auxiliary Data Retrieval

Figure 6 shows in more detail, how auxiliary seismic data retrieval is accomplished. It can be seen that components of the DACS Subsystem, the Retrieve Subsystem and the Message Subsystem cooperate to this effect. See [IDC7.4.3] and [IDC6.5.20] for details of design and use, respectively.

Figure 6 Auxiliary data retrieval path on the IDC Development LAN
3.2.7.5. **LEB Simulation and Post-Analysis Processing**

The Post-analysis processing on the Development LAN is implemented in a different way compared with the IDC Testbed or Operations LANs.

The *auto* crontab file on the main Dev LAN processing server contains an entry for the script `~auto/bin/leb_simulator` to simulate interactive LEB data by copying the automatically created SEL3.

If the SEL3 bulletin exists for a complete day the LEB simulator script will run the `mig_bull` script to copy the SEL3 bulletin to the LEB database account for the entire day. The script also inserts entries into the LEB.*allocate_hour* table. Insertion of the entries with actions `Del_Pass` and `Scan_Pass` triggers recall processing via the standard database trigger between the LEB .*allocate_hour* table and the IDCX .*interval* table.

The *leb_simulator* script then waits for recall processing to complete. Then the remaining *RebDone* entries are inserted into the LEB.*allocate_hour* table. This action then triggers the post-analysis processing pipelines. If the script is manually killed during this waiting time, the post-analysis pipelines will not be triggered for the copied bulletin.

The *leb_simulator* script is activated and deactivated by editing the corresponding entry in the *auto* user’s crontab file. It writes a log file to the daily log directory for `mig_bull`: `$(LOGDIR)/<jdate>/mig/sel3toleb=<machine>=autocron`. 
3.3. Operations Procedures

3.3.1. Starting the SHI Processing System
To start SHI data processing the following procedures have to be followed:

- Login to the processing servers as auto and cmss in two separate windows using the ssh command.
- Start the CDTools, DACS and Message/Subscription Subsystems following the start up procedures for these subsystems (CDTools as cmss, others as auto).
- Monitor the system to verify a successful system start following the monitoring procedures in section 3.4.

3.3.2. Starting the CDTools and cd2w

Abstract:

CDTools and cd2w the principal applications in the IDC data acquisition system must be started to allow data acquisition. They are independent from the rest of the data processing system and partially independent of the database. This design feature increases robustness of the system and allows receiving and storing incoming data even if there is a database problem or the automatic processing fails completely.

Additional information: CDTools Software User Guide, see [IDC-CDT-SUG].

Procedure:

- Login as cmss on the CDTools processing server.
- Start each individual instance of the CDTools (cdrecv, cdsend, cd2w) for each of the three ports that are used on the Development LAN. It is useful to create Unix aliases for each command below, where the <hostname> is to be replaced by the name of the processing machine:

```
$(CMS_SCRIPTS)/bin/start.cdtool -p $(RELBIN)/cdrecv -c $(CONTINUOUS_DATA-DIR)/cdtools/cdtools.$port.par -l $(LOGDIR)/cdtools -o cmss -h <hostname> -u cmss &
```

```
$(CMS_SCRIPTS)/bin/start.cdtool -p $(RELBIN)/cdsend -c $(CONTINUOUS_DATA-DIR)/cdtools/cdtools.$port.par -l $(LOGDIR)/cdtools -o cmss -h <hostname> -u cmss &
```

```
$(CMS_SCRIPTS)/bin/start.cdtool -p $(RELBIN)/cd2w -c $(CONTINUOUS_DATA-DIR)/cdtools/cd2w.$port.par -l $(LOGDIR)/cdtools -o cmss -h <hostname> -u cmss &
```

where $port can be take the values 8000, 8100 and 8200.

- Check the logs to verify that all instances have started successfully and are running. Note that for debug=0 the log files for cdrecv and cdsend will be empty and cd2w will only log the start time and its version number.
cd $(LOGDIR)/cdtools
ls -ltr

- Verify that the timestamp of the last log file is the same or later as the time when the CDTools were started. Also check the syslog file
  /ctbto/syslog/$(/lan)/yyyy/mm/dd/CTBT, where yyyy/mm/dd stands for the
  4-digit year, and 2-digit month and day.

- Check that new waveform files are being written since the cd2w have been started:
  cd $(SHI_DATADIR)/cds/waveforms/<yyyy>/<mm>/<dd>
  ls -ltr

  Compare the timestamp of the last written file to the current date and time.

- Query the wfdisc table in the IDCX database account to verify that new entries are
  inserted:

  sqlwrap IDCXDB
  select max(wfid) from wfdisc;
  select * from wfdisc where wfid=<wfid>;
  select to_char(lddate,'DD-MON-YYYY HH:MI:SS') from wfdisc \
       where wfid=<wfid>;
  quit;

3.3.3. Starting the Message and Subscription Subsystems

Abstract:
In order to start the Message and Subscription Subsystems the following procedures have to
be followed:

Additional information:
[IDC6.5.19] Message Subsystem Software User Manual,
[IDC6.5.20] Retrieve Subsystem Software User Manual,

Procedure:
- Login as auto on the processing server for the Message and Subscription Subsystems.
- Uncomment the crontab entries for keep_subs_alive and keep_msg_alive:
  crontab -e

  Either wait for cron to run or call the scripts keep_subs_alive and keep_msg_alive
  manually to start all related applications immediately:
  keep_msg_alive
  keep_subs_alive

- Monitor log MessageReceive and MessageGet log files to identify problems if there
  are any.
3.3.4. **Starting the DACS**

**Abstract:** The Distributed Application Control System (DACS) coordinates the execution of all automatic processing applications, making use of components of the commercial middleware *Tuxedo* and custom-developed applications that access *Tuxedo* components. All automatic processing pipelines require *Tuxedo* to be properly installed and running. The start-up procedure has multiple steps, which are described below.

**Additional Information:**

**Procedure:**

- Login to the *Tuxedo* master host as `auto`:

- Make sure that no DACS processes are currently running:
  ```
  ps -fu auto
  ```

- Clean old shared memory resources on all DACS hosts. Check for unused shared memory:
  ```
  usr_ipc
  ```
  If the result is non-zero, remove shared memory resources:
  ```
  usr_ipc -DF
  ```

- Check that the *tlisten* process is running on all DACS hosts:
  ```
  ps -fu auto | grep tlisten
  ```

- Start *tlisten* locally on all DACS hosts if it is not running:
  ```
  /var/tuxedo/IDC3_process/bin/start_tlisten
  ```

- Start *Tuxpad* as user auto on *Tuxedo* master host:
  ```
  start_Tuxpad
  ```

- Start the DACS from *Tuxpad*:

- Bring up the *Tuxpad* message window and check for any errors during the following steps.

- Boot SysAdmin processes the primary and backup *Tuxedo* master host.

- Boot Application processes on all DACS hosts.

- Refresh the *Tuxpad* window and check if all machines, groups and DACS servers have turned green.

- Bring up the *Qinfo* and *Scheduler* windows and check that the system starts queuing and scheduling. The schedule queue should hold exactly one element which is the scheduler table. If the schedule queue is empty press the Re-init button in the *Scheduler* window to initialize the scheduler. After this action automatic processing will start immediately.
Stall or un-stall pipelines from the Scheduler window as needed. If the system has to catch up with processing it will typically be preferable to stall the network processing pipelines until station processing has caught up with almost real time. The delay value in the Scheduler window can be used to schedule an un-stall command for a later time (e.g. overnight).

Start WorkFlow and verify that the system starts processing any queued intervals:
```
start program=WorkFlow
```

In case there are no new intervals displayed by the application, check if QSpace contains any new intervals.

Start RequestFlow and verify that any new requests are dispatched.
```
start program=RequestFlow
```

### 3.3.5. Starting the Interactive Analysis Software

**Abstract:**
Interactive analysis software includes various applications to analyze SHI data. These are ARS, XfkDisplay, Map, Geotool, etc. Each of them can be started as a standalone application. However in order to communicate to each other they depend on Tuxedo (or nontux in the future) to transmit IPC messages. Instructions on how to start the interactive software applications in an intercommunicating mode are given below.

**Additional information:**

**Procedure:**
- Login to one of the interactive analysis workstations as auto.

- Set the CMS_MODE to analysis and source the related system environment:
  
  ```
  setenv CMS_MODE analysis
  source ~/.cshrc
  ```

- Check if any shared memory resources are left over and remove them:
  ```
  usr_ipc
  usr_ipc -DF
  ```

- Check if all 16 interactive Tuxedo processes are running and start them if needed:
  ```
  ps -fu auto
  tmboot –y
  ```

- Use the ps command to check which processes are running. Currently, 16 individual Tuxedo processes are expected.

- Repeat the above steps for other interactive analysis machines as needed.

- Login to the same interactive analysis workstation as SHI analysis user.

- Check the CMS_MODE variable setting, set it to analysis if needed and source the system environment:
printenv CMS_MODE
setenv CMS_MODE analysis
source ~/.cshrc

- Start interactive analysis tools:
  start program=analyst_log

- Start **dman** by pressing the ARS button in **analyst_log**. Alternatively start **dman** on the command line:
  start program=dman

- Start **ARS** and other analysis tools from **dman** as needed.

### 3.3.6. Starting the Web Subsystem

**Abstract:**
Generally the IDC Web Subsystem is the main interface to the external users. However considering that the Development LAN is a testing area, the Web Subsystem does not play that important role. It is commonly used for testing bug fixes and software updates only.

**Procedure:**
- Login to the web host as **webcm** on the Dev LAN (**webadm** on other IDC LANs).
- Verify that the Netscape web server is running as **webadm**:
  
  ps -fu webadm

- The expected process on the Dev LAN is (the server configuration was once copied from kuredu, thus the naming):
  
  ns-httpd -d /opt/netscape/https-kuredu/config

- Contact the web administrator or a person having **webadm** privileges to restart the web server if needed.
- Verify that the **webcm cron** jobs are activated and uncomment the **crontab** entries if needed.
- Start daily web updates:
  
  cd $(lan)/software/shared/web/web-content
  make daily_updates

- Open the IDC products web page at [http://eldey.idc.ctbto.org](http://eldey.idc.ctbto.org). IDC products should be available on the web after both database entries for the data products and the interval table have been migrated to the archive database. This will typically be the case a few hours after creation of the data products. Also check the status of data migration (section 3.4.4.)

### 3.3.7. Shutdown of the SHI Processing System

**Abstract:** Shutting down SHI data processing system is a common task which takes place often due to various software changes or system failures.
Additional information:

Procedure:
- Login to the relevant processing servers as auto and cmss in two separate windows.
- Shut down the DACS, Message/Subscription Subsystems and CDTools following the shut down procedures for these subsystems.
- Check the crontab entries for cmss on the server for the Message and Subscription Subsystems to make sure that no keep_alive cron jobs are left active and will restart Message or Subscription Subsystem processes. The remaining cron jobs can be left active for a normal system shut-down. However, deactivate them if there is a special reason that the corresponding processes should not be running while the system is down (e.g., deactivate MigrateData if the database is to be shut down, or deactivate Archive if the mass store system is not available).
- Check with ps -fu cmss and ps -fu auto that no unexpected processes are left running. There should be no running processes left for cmss. For auto the Tuxedo tlisten process and the tail process copying the current Tuxedo ULOG file are expected to be left running on the primary and backup Tuxedo masters. They do not normally need to be shut down but may be forced to exit using the kill command if desired.

3.3.8. Shutdown of the DACS

Abstract: Shutting down the DACS is a common task which takes place often due to various software/hardware changes or system tests/failures.

Additional information:

Procedure:
Login to the Tuxedo master as user auto.
- Start Tuxpad (if not already running)
  start_Tuxpad

In Tuxpad:
- Open the message window.
- Shutdown all DACS processes using the button "Shutdown All" and observe shut down messages in the message window.
- Refresh the Tuxpad window and wait for all DACS machines, groups and servers to be shown in red.
- Exit from Tuxpad.
- Cleanup left-over shared memory resources on the primary and backup Tuxedo master hosts.
  usr_ipc -DF
3.3.9. **Shutdown of the Message and Subscription Subsystems**

**Abstract:**
The Message and Subscription Subsystems often require reboot due to software updates or changes.

**Additional information:**
[IDC6.5.19] Message Subsystem Software User Manual,
[IDC6.5.20] Retrieve Subsystem Software User Manual,

**Procedure:**

As user `auto` on the server for the Message and Subscription Subsystems:

- Disable `cron` job entries `keep_msg_alive` and `keep_subs_alive` in the `auto` crontab file. Otherwise, the processes will be automatically restarted.

- Shutdown the following processes using the command `kill`: WaveAlert, MessageGet, MessageReceive, SubsProcess, MessageShip:

  ```
  ps -fu auto | egrep -i mess
  ps -fu auto | egrep -i subs
  kill <PIDs>
  ```

  Alternatively use the `pkill` command with the process names as arguments:

  ```
  pkill MessageGet MessageReceive MessageShip WaveAlert SubsProcess
  ```

- Verify that no processes are left over:

  ```
  ps -fu auto | egrep "WaveAlert|MessageGet|MessageReceive|\ SubsProcess|MessageShip|AutoDRM"
  ```

- If any processes are still running retry the `kill` command. If this is again unsuccessful use the `kill -9` option to force the process to terminate.

3.3.10. **Shutdown of the CDTools and cd2w**

**Abstract:**
CDTools and cd2w may require shutdown when there is a need to stop data acquisition from all stations or a particular subset of them.

**Additional information:**

**Procedure:**

- Login to the processing server for the CDTools as `cmss`.

- Shut down the `start_cdtool` script instances. The PIDs for each instance can be found in `cdtools` log directory as part of the log file names.

  ```
  cd $(LOGDIR)/cdtools
  ls -ltr
  ```
The naming convention of the log files is <program>.<port>.<PID>.log. Kill the processes listed in the filenames.
`kill -9 <PID>`

- Shutdown CDTools after the start scripts (parent processes) have been terminated.

```bash
ps -fu cmss | egrep "cd2w|cdsend|cdrecv" | grep -v grep
kill <PIDs>
```

- Verify that neither CDTools nor any of the related `start.cdtool` processes are running.

```bash
ps -fu cmss | egrep 'cd2w|cdsend|cdrecv|start.cdtool' | \ grep -v grep
```

### 3.3.11. Shutdown of the Interactive Analysis Software

**Abstract:** Interactive analysis tools can be exited individually by pressing the respective buttons. Shutdown of the interactive DACS processes is described here.

**Additional information:**
[IDC6.5.1] Interactive Analysis Subsystem Software User Manual

**Procedure:**
- Exit from all interactive analysis tools.
- Login to one of the interactive analysis workstations as `auto`:
- Set the `CMS_MODE` to `analysis` and source the related system environment:
  ```bash
  setenv CMS_MODE analysis
  source ~/.cshrc
  ```
- Shut down all interactive `Tuxedo` processes, watch messages in the window during the shut down process and check if any interactive `Tuxedo` processes are left running:
  ```bash
  tmshutdown -y
  ps -fu auto
  ```
- Check if any shared memory resources are left over and remove them:
  ```bash
  usr_ipc
  usr_ipc -DF
  ```
- Repeat the above steps for other interactive analysis machines as needed.

### 3.3.12. Shutdown of the Web Subsystem

**Abstract:** This is the procedure for terminating the daily web updates and the web server process.

**Procedure:**
- Login to the web host as `webcm` on the Dev LAN (`webadm` on other IDC LANs).
- Deactivate `webcm cron` jobs.
• Remove the currently scheduled `at` job for the daily web updates:
  
  `at -l`
  
  `at -r <at_job_id>`

• Contact the web administrator to shut down the web server using the *Netscape enterprise server* "admin" interface [http://eldey.idc.ctbto.org:5588](http://eldey.idc.ctbto.org:5588) for the Dev LAN if specifically required.

• Check if the web server is running:
  
  `ps -fu webadm`

### 3.3.13. Starting and Stopping Cron Jobs

To shut down all `cron` jobs for the entire system login as `auto`, then `cmss` and `webcm`, and open the `crontab` file for each user:

```
crontab -e
```

Add the first character `#` to all `crontab` entries to comment the entries out, then exit from the `crontab` editor, or alternatively use the following command to automatically comment out all `crontab` entries:

```
crontab -l | sed s/^/#/#/ | crontab
```

No `cron` jobs will be started for the current user after exiting from the `crontab` editor. In case of an OS upgrade `crontab` entries usually get deleted. Therefore it is useful to have them saved somewhere, for example under `ClearCase`.

To start cron jobs it is necessary to remove `#` characters from the beginning of relevant lines.

### 3.3.14. Clean-up Cron Jobs

The `auto crontab` file on the main Dev LAN processing server contains an entry for the clean up script `~auto/bin/purge_data`. It is called with a number argument defining the number of days after which old data shall be purged. The argument is currently set to 15; thus the script purges data directories, which are 15 days old. To retroactively purge older data directories the script can be run manually with different arguments.

Since SHI processing products are not archived for the Development LAN the `purge_data` script also deletes old products. The complete list of data and products, which are deleted for the given day, is:

- auxiliary data
- messages
- segment archive data
- threshold monitoring products

The script also deletes database entries corresponding to the removed data and products files in the following tables (all in `IDCX@idcdev`, except where specified otherwise):

- `wfdisc`
- `wftag`
The purge_data script is activated and deactivated by editing the corresponding entry in the auto crontab file on the main Dev LAN processing server. It writes a log file to the daily log directory also used by MigrateData:

\$(LOGDIR)/<jdate>/MigrateData/purge_data.log.

The crontab file for cmss on the the main Dev LAN processing server contains entries to remove old frame and waveform files which are generated by the CDTools in the directories \$(SHI_DATADIR)/cds/frames/receiver and \$(SHI_DATADIR)/cds/waveforms as well as old CDTools log files in the directory \$(LOGDIR)/cdtools. The entries contain the entire commands and no script is used in this case. These crontab entries should be deactivated if incoming frame and waveform files are archived by the Archive application.

The need for other clean up cron jobs may come up temporarily on the Development LAN to implement workarounds for similar problems or to operate new software that has not yet been fully integrated into the existing processing system. Depending on the owner of the files to be removed by such cron jobs, the entry will be in the auto or cmss crontab file on the main Dev LAN processing server. Check both crontab files to determine if such additional cron jobs are activated at any given time.

3.3.15. Controlling Data Migration and Purging

Abstract: This section describes the procedures to control the migration of data from the operational database to the archive database and the purging of old data from the operational database.

Additional Information: See section 3.4.4. for functionality and correct configuration of Migration and Purge.

Procedure:

crontab entries for data migration and purging are in the crontab file for cmss on the the main Dev LAN processing server. They start and control a series of migration and purge instances of the application MigrateData in order to migrate data from the operational to the archive database.

- Open the crontab file for cmss:
  
  crontab -e

- To find data migration cron jobs look for cron_migrate class=mig entries in the crontab file.

- To find purging cron jobs look for cron_migrate class=purg entries in the crontab file.

- Comment or uncomment the relevant entries using the # character.
Exit from the `crontab` editor. The changes will become effective at this point.

### 3.3.16. Controlling Data Archiving

**Abstract:** This section describes how data archiving can be simulated with the Development LAN.

**Additional information:** [IDC6.5.22] Archive Subsystem Software User Manual.

Data are not archived permanently on the Development LAN, due to limited storage space in the file system. The Development LAN `archive` directory is not located on the mass store but in the standard file system. Thus `Archive` is run only for software testing, and the archived data are deleted after a few days.

**Procedure:**

All archiving `crontab` entries are in the `cmss crontab` file on the main Dev LAN processing server. There are `cron` jobs to start the `Archive` application for various archiving classes, to check the success of data archiving, and to create new archiving intervals in the `interval` table.

- To change the status of archiving `cron` jobs:
  
  Open the `crontab` file for `cmss`:
  
  ```bash
  crontab -e
  ```

- To find archiving `cron` jobs check for the following strings in the `crontab` file:
  
  ```bash
  run_archive, Check_arch, AUXWF_create.
  ```

  Comment or uncomment as needed and exit from the `crontab` editor.

- To change the status of archiving `cron` jobs open the `crontab` file for `cmss` and comment or uncomment the entries of interest.

### 3.3.17. Controlling keep_alive Scripts

**Abstract:**

This section describes how to control `keep_alive` scripts. Such scripts monitor the continuously running applications of the Message and Subscription Subsystems and restart them in case of unexpected failure.

**Additional information:**

[IDC6.5.19] Message Subsystem Software User Manual,

**Procedure:**

There are two "keep alive" `cron` jobs for `auto` to automatically restart the Message Subsystem and Subscription Subsystem processes.

To (de)activate the message and subscription "keep alive" `cron` jobs:

```bash
   crontab -e  #edit the crontab file, save and exit
```
Search for the keep_msg_alive and keep_subs_alive entries and (un)comment them as needed, then exit from the crontab editor.

3.3.18. Controlling Web Subsystem Cron Jobs

Abstract:
All crontab entries for the Web Subsystem are in the crontab file for webcm. Their purpose is to update specific web pages, which are not included in the daily update mechanism scheduled by the daily at job.

Additional information:

Procedure:
To activate or deactivate cron jobs for the Web Subsystem on the web host:

crontab -e

Comment or uncomment the relevant crontab entries, then exit from the crontab editor.

3.3.19. Controlling Other Cron Jobs

Abstract:
There are further crontab entries in the crontab files for auto and cmss on the main Dev LAN processing server, which control a number of additional processes, most of which are regularly run to clean up specific data files or database tables in the system. They can be activated and deactivated using the same procedure as for other cron jobs. Make sure to look in the correct crontab file. Individual crontab entries and their purposes are listed below.

Procedure:

crontab for auto:

purge_data: Dev LAN specific script to purge various data directories and related database tables after a configurable number of days.

leb_simulator: Dev LAN specific script to migrate the SEL3 database account to the LEB account for a full day when network processing in SEL3 is complete for that day. The script also copies entries to the allocate_hour table which will trigger recall processing and, after a configurable delay, post-analysis processing. It is used to simulate an LEB and REB on the Development LAN without the need for interactive analysis to be done by human analysts. See section 3.2.7.5. for more specific information on the leb_simulator script.

delete_beams: Purges beam waveforms and related database tables after a configurable number of days.

delete_dervdisc: Purges the dervdisc table used by DFX-pmcc as well as related data files.

clean_IMStables: Removes temporary database tables used by ARS during interactive analysis.
polling: Requests a one-minute interval of data from all auxiliary stations listed in the sitepoll table.

pruneque: Purges the Tuxedo queues "failed" and "failed-dispatch".

logswitch_midnight: Collects the local Tuxedo ULOG files from each automatic DACS host and merges them to a central CLOG file on THOST. Starts local tail jobs on all automatic DACS hosts to collect the ULOG file for the next day.

cronmklog: Creates the log directories for the next day. Automatic processing will stop if this cron job does not run and the log directories for today or for earlier days, which are processed in the network processing and post-analysis pipelines, do not exist.

crontab for cmss:

run_squal: Several crontab entries to start a number of squal instances which generate daily station capability reports.
3.4. Monitoring Procedures

3.4.1. Routine Monitoring Tasks

Abstract:
Although system monitoring is primarily done by the system administrator the commands and tools described below are recommended to check the availability and performance of the Development LAN servers and databases.

Additional information: All IDC Software User Manuals (SUM) have a section “Monitoring” in their “Troubleshooting” chapters.

Procedure:
The following links can be used to monitor the status of various system parameters for Development LAN machines:
http://eldey.idc.ctbto.org:1983
http://flatey.idc.ctbto.org:1983
http://dls001.idc.ctbto.org:1983
http://dls003.idc.ctbto.org:1983

- A disk usage report is generated automatically every day between 00:00 and 00:30am for all relevant file systems on Development LAN, Testbed and Operations LANs and available under the following URL:
  http://eldey.idc.ctbto.org:8888/Quotas/

- As these statistics are collected only once per day the following commands are useful to get the current status:
  du -hs  #summarizes disk usage for current directory
  df -h   #displays disk usage for entire file system

- Examine active processes on a system and report statistics:
  top     #displays the top 15 processes on the system as well as memory, CPU and swap space usage
  prstat  #reports active process statistics. For example,
           #prstat -s cpu -t reports a total CPU usage
           #summary for each user sorted in descending order

- Check database availability:
  ping    #tests connectivity to a remote machine
  tnsping #checks if the Oracle database listener is up.
           #Usage: tnsping <net service name> where net #service name is the service name of database
           #instance. Example: tnsping idcdev

3.4.2. Monitoring the Automatic Processing Workflow

Abstract:
WorkFlow is the main monitoring tools for the automatic processing software. It should be running constantly on the workstation of the Development LAN operator in order to allow timely indication of processing problems.
Additional information:
[IDC6.2.1] Release 2 Operations and Maintenance Seismic, Hydroacoustic, and Infrasonic System,
[IDC6.5.2] Distributed Application Control System (DACS) Software User Manual,
Monitoring sections in other SUMs

Procedure:
- Login to any machine on the Development LAN. The WorkFlow interval reprocessing feature (see bullet 4 below) will only be available for the user auto on the primary and backup Tuxedo master hosts:
  ```
  start program=WorkFlow
  ```
- Check WorkFlow for any red (i.e. failed) intervals. WorkFlow shows a graphical representation of the interval table in the database; thus, querying the interval table is an alternative method to check the processing status.

Red intervals point to failures of specific applications to process data from these intervals. Clicking the right mouse button on the interval will show the interval id, the start and end time of the interval and the interval status. In combination with the interval class and name values shown on the left side of WorkFlow these data identify an entry in the IDCX.interval table of the database. The interval status also gives a hint on the specific application which failed.

- Check the corresponding application log file for error messages:
  ```
  cd $(LOGDIR)/<jdate>/<application-logdir>
  vi <application-logfile>
  ```
  The tuxshell application writes to the current <jdate> directory at run time. Most applications write to the <jdate> directory for the data time of the processed interval.
- If the error condition was transient or the inspection of the log file leads to an error that can be immediately resolved, reprocess the interval from WorkFlow after the error condition has been removed. Right-click the failed interval in WorkFlow and select the Process tag in the menu to reprocess the interval. The reprocessing function only works if WorkFlow was started on the primary or backup Tuxedo master host as auto user.
- If the error condition is unclear or cannot be resolved refer to section 3.6.5. for further procedures.
- Check WorkFlow for delayed or hanging processing pipelines.
- The TI/S station processing pipelines should be close to real-time for all stations which are currently received by the CDTools. An exception to this normal operating status is a catch-up situation when the system has been restarted after an extended shut down time. In a catch-up situation a large number of yellow (queued) station processing intervals can be expected after the CDTools have been restarted, and stations will usually back-fill. In this case, current data intervals will be processed with a higher priority than back-filled data.
If data are received by the CDTools but not processed, check the scheduler table using Tuxpad to see if the station processing pipeline (tis, tis-late, tis-verylate) is stalled and unstall it with a short delay value if necessary.

The sel1, sel2, sel3 network processing pipelines are scheduled with delays of 1:20 hours, 5:20 hours and 11:20 hours behind real time. Check the scheduler table, if the delays seem unusual, and ensure that the pipelines are unstalled.

The Recall and post-analysis pipelines depend on the leb_simulator cron job. The leb_simulator script is configured to copy the SEL3 to the LEB database account once the sel3 bulletin is complete for a full calendar day. This results in a 1-2 day delay of Recall and post-analysis processing with respect to real time. If this is not the case check the auto crontab entry and the log file in the logs/<jdate>/mig directory to see if the cron job has run and if there was any problem with copying the sel3 bulletin. Activate the cron job if it is deactivated.

If all pipelines are hanging at the same time and there is no DACS processing anymore, there is probably a general problem in the system rather than an application-specific issue. Check if the log directories for today and the last few days exist and are read- and writable. Create any missing log directories using the mklogdir script. If there is no apparent reason for the problem consider shutting down all DACS processes, cleaning the shared memory resources using usr_ipc -DF and restarting the DACS.

3.4.3. Monitoring the Data Request Workflow

Abstract:
RequestFlow is an application which allows monitoring of requests to auxiliary stations and their success. It is a useful tool to monitor the Retrieve Subsystem.

Additional information:
[IDC6.5.19] Message Subsystem Software User Manual,
[IDC6.5.20] Retrieve Subsystem Software User Manual,
[IDC7.1.11] WaveExpert
[IDC7.4.3] Retrieve Subsystem.

Procedure:

• Login to any machine on the development LAN:
• Start RequestFlow:
  start program=RequestFlow
• Check the status of data requests in the Retrieve Subsystem. RequestFlow shows a graphical representation of the request table in the database; thus, querying the request table is an alternative method to check the request status:
• If there are no recent data requests, check the processing status of the sel1, sel2, sel3 and lp pipelines in WorkFlow. These four pipelines include instances of WaveExpert which automatically create new requests. The sel1, sel2 and sel3 instances of WaveExpert will only create requests if network processing was able to create any events. This will typically not be the case if there are little or no data are received by the CDTools or if station processing is stalled or shut down.
• If there are new requests but they remain in status requested, check the status of the WaveGet data monitor in the scheduler table using Tuxpad and unstall WaveGet if it is stalled.

• If requests are in status dispatch-failed check the dispatch log files for error messages to find and resolve the error condition. If requests remain in status running this means they have been successfully submitted to the IDC Email system and the IDC data processing system is waiting for data messages from the stations. There may be problems at the station side or with routing emails through the network. Check the Message Subsystem to verify if there are any incoming messages and if they are correctly processed (see section 2.4.8). Contact the appropriate contact person to verify if the recipients of the request messages have received the sent requests.

• If requests are in status done-success data has been received from the stations and station processing intervals should be queued in the interval table. Use WorkFlow to monitor the status of station processing for data received from auxiliary stations. Also the status can be done-partial which means that at least some of the requested data was received. Usually this is an indication of the problem at the station side.

• There is no manual retry feature in RequestFlow. The Message Subsystem application WaveAlert automatically compares the status of the request table with received data in the wfdisc table and updates the request status to done-success if all data has been received. If no data are received WaveAlert will update the request status to retry after a configurable time and the request message will be sent again. WaveAlert will update the request status to failed if there is still no data response after a second configurable time interval. Thus, contrary to WorkFlow, the request status failed does not indicate that an application has failed to process data, but it indicates that no data has been received and parsed for this request.

3.4.4. Monitoring the Data Migration WorkFlow

Abstract:
MigrateFlow is the most useful tool for monitoring data migration and purging. Though it does not allow reprocessing of failed intervals from the graphical interface it detects and displays them quickly and efficiently.

Procedure:
• Login to any machine on the development LAN as auto.

• Start the MigrateFlow
  WorkFlow par=$(CONTRIB_HOME)/par/MigrateFlow.par &

• Check the status of data migration and purging. MigrateFlow shows a graphical representation of the MIGRATE.interval table in the database, thus querying the MIGRATE.interval table is an alternative method to check the data migration and purging status. Both data migration and purging are done by the MigrateData application, which runs for various database accounts and tables from cron. All relevant entries are in the crontab file for cmss on the main Dev LAN processing server. Each cron job corresponds to one line in MigrateFlow.
Data migration is configured to run 1-2 hours behind real time. The *cron* jobs for data migration migrate all data for particular tables or database accounts from the main database (*idcdev*) to the archive database (*idcdev2*).

Purging of data is configured to run 15 days after real time (42 days on Testbed and in Operations). For each data migration *cron* job there is a purge *cron* job which compares the original with the archived data and then deletes the original data in the main database (*idcdev*). Purging will fail if there are any differences found between the original and archived data.

If any migration or purge jobs are unusually delayed, check if the corresponding *cron* jobs are activated.

If there are any failed intervals (shown in red) check the *MigrateData* log files for the current day. In almost all cases the reason for purging failures are differences that were found between the original and archived data. The log files will indicate an auxiliary table in the migrate database account which points to the particular differences. Remove the differences by either deleting the additional entries in one database or by copying these entries to the other database. Run the *cron* job again to check if additional differences are found or if it runs successfully.

An alternative method to work around failed *MigrateData* intervals is to update the `last_mig_date` column in the `MIGRATE.mig_date` table for the particular *cron* job (identified by the `class` and `procname` values) and set it to a time beyond the problem interval. The next *MigrateData* run will then continue from the new `last_mig_date` time, ignoring the failed interval.

### 3.4.5. Monitoring the Data Archiving WorkFlow

**Abstract:**

Though data is not archived permanently on the Development LAN, there is still a need for a monitoring tool to reflect the status of the archiving processes. The *WorkFlow* application with a specially constructed parameter file can be used to display the data archiving status.

**Additional information:**

[IDC6.5.22] Archive Subsystem Software User Manual,
[IDC7.5.1] Archiving Subsystem.

**Procedure:**

- Login to any machine on the development LAN.
- Start *WorkFlow* for file archiving:
  ```
  WorkFlow \
  par=$(CMS_CONFIG)/app_config/distributed/WorkFlow/WorkFlow_arch.par &
  ```
- Check the file archiving status. The *Archive* application is not routinely run on the Development LAN, thus it only needs to be monitored during dedicated archiver tests. The *Archive* application writes entries to the *IDCX.interval* table using specific class values to identify archiving intervals. The *WorkFlow* for file archiving shows a graphical representation of the archiving intervals. Querying the interval table is thus an alternative method to check the archiving status.
Check the status of the cron jobs for run_archive, Check_arch and AUXWF_create to determine which types of files should currently be archived. If there are no archiving intervals or failed intervals for active archiving cron jobs, check the corresponding log files to determine the particular problem.

AUXWF_create creates new archiving intervals for auxiliary stations, including continuously requested auxiliary stations, which are archived with class PRIARC.

All other new intervals are created by the Archive application, which is started by run_archive. The Archive application is responsible for queuing and archiving all new intervals at the configured times, depending on the archiving class.

Check_arch compares the original data with the archived data and updates the intervals to VERIFIED or NOTVERIF, depending on the result of the comparison.

### 3.4.6. Monitoring the CDTools and cd2w

**Abstract:**
Monitoring CDTools and cd2w is primarily done by monitoring CDTools processes, data files and log files.

**Additional information:**
CDTools Software User Guide, [IDC-CDT-SUG]

**Procedure:**
- Login as cmss on the CDTools processing server.
- Verify that all CDTools and their corresponding start.cdtools instances are running:
  ```bash
  ps -fu cmss | egrep 'cd2w|cdsend|cdrecv|start.cdtool' | grep -v grep
  ```
- Verify that frame files and waveform files are being written:
  ```bash
  ls -ltr $(SHI_DATADIR)/cds/frames/receiver/<yyyy>/<mm>/<dd>
  ls -ltr $(SHI_DATADIR)/cds/waveforms/<yyyy>/<mm>/<dd>
  ```
- Check the CDTools log files and the syslog file for error messages:
  ```bash
  cd $(LOGDIR)/cdtools
  ls -ltr
  vi <logfile>
  ```
- If stations do not connect and no local problem can be found, contact the sender of the data to determine if there is a problem at the sending side or with the connection to the IDC.
- If cdsend is unable to connect to data recipients, contact the data recipient to determine if the recipient is ready to receive continuous waveform data.
3.4.7. Monitoring the DACS

Abstract:
Tuxpad is the main tool to monitor DACS processes.

Additional information:
[IDC6.5.2] Distributed Application Control System (DACS) Software User Manual,
[IDC7.3.1] Distributed Application Control System (DACS).

Procedure:
• Start Tuxpad as auto on the primary or backup Tuxedo master host:
  start_Tuxpad

• Verify that all DACS machines, groups and servers are running (shown in green on Tuxpad).

• Check the scheduler table in the Scheduler window to verify that all pipelines are unstalled and that the next wakeup times are reasonable. Most wakeup times should be within the next hour. An exception is tis-verylate, which is scheduled to wake up every twelve hours.

• Check the queuing status in the Qinfo window. The schedule queue should always hold exactly one element, which is the scheduler table. For other queues the sum of all messages should not exceed 50000.

• If the schedule queue is empty, all DACS processing will be halted. Re-initialize the scheduler from the Schedule window to generate a new scheduler table in the schedule queue.

• If there are elements in the queues failed or failed-dispatch, there are failed intervals in WorkFlow, respectively the dispatch script failed to generate and send automatic data requests. Check WorkFlow (for dispatch failures check RequestFlow) to determine which application has failed to process which data interval(s). Check the appropriate application log files and the corresponding tuxshell log files to further determine the specific problem(s).

• Note that the application and tuxshell log files for the same data interval may be in log directories for different days as follows:
  $(LOGDIR)/<data_jdate>/<app_logdir>/<app_logfile>
  $(LOGDIR)/<runtime_jdate>/tuxshell/<app>-<machine>-<pid>
  /var/tuxedo/IDC3_process/ULOGS/<jdate>

• If there are any scheduling or queuing problems, check the log files for the data monitors and other internal DACS processes in the following directory:
  $(LOGDIR)/<runtime_jdate>/dacs

• Contrary to the done and done-dispatch queues, the failed and failed-dispatch queues are not automatically purged. Use the qmadmin command line interface as auto on the Tuxedo QHOST to purge these queues after the problems with the failed intervals have been addressed:
qmadmin
qopen DACS
qset failed
qdltm -y
qset failed-dispatch
qdltm -y
qinfo
quit

3.4.8. Monitoring Message and Subscription Subsystem Processes

Abstract:
Though Message and Subscription Subsystems are quite robust, they still require monitoring checks. They are described below.

Additional information:
[IDC6.5.20] Retrieve Subsystem Software User Manual,
[IDC6.5.21] Subscription Subsystem Software User Manual,
[IDC7.4.2] Message Subsystem,
[IDC7.4.4] Subscription Subsystem.

Procedure:
• Verify that the continuously running message and subscription processes are indeed running:
  ps -fu auto | egrep -i "mes|subs"

• Continuously running processes are: MessageReceive, MessageGet, MessageShip, WaveAlert, SubsProcess.

• If some or all of the above processes have died, check if the keep_msg_alive and keep_subs_alive crontab entries for auto on the processing server for the Message and Subscription Subsystems are activated. Also, the mailbox of auto user should contain messages indicating recurring restarts of applications relevant to the Message or Subscription Subsystems.

• Activate the two cron jobs if needed and wait for cron to automatically restart the message and subscription processes, or execute the keep alive scripts from the command line to immediately restart the processes.

• Check the message and subscription application log files in $(LOGDIR)/msg to determine if messages are sent and received as expected and to check for error messages and/or failures.

• Check RequestFlow to determine the status of automatic data requests which are generated by the Retrieve Subsystem.

• Check the IDCX.msgdisc table to determine if there are any failed messages or messages which are left in status RUNNING after an application failure. Messages which are left in status RUNNING will block the internal message queues. If one or more message queues are blocked there will be a large number of messages in status RECEIVED, which have been received and are waiting to be processed.
The message queues are configured in the MessageGet parameter file and are used to queue incoming messages for further processing based on the message type. If incoming data messages are not being parsed the ParseData message queue may be blocked. If incoming request messages are not being processed, the corresponding AutoDRM message queue may be blocked. In both cases find the blocking message in status RUNNING in the IDCX.msgdisc table and update the status either to RECEIVED in order to reprocess the message, or to FAILED in order to avoid reprocessing of the particular message.

If MessageReceive does not process any incoming messages check if there are messages waiting in the temporary storage directory $(MSGDIR)/msg_TSD and if there are rejected messages in $(MSGDIR)/invalidmail.

If messages accumulate in the msg_TSD directory check if MessageReceive is running and correctly configured.

If messages are found in the invalidmail directory there may be a problem with message authentication. Check if the senders of the message have valid entries in the IDCX.datauser and IDCX.subsuser tables in the database, and check the authentication configuration of MessageReceive.

If no messages are received at all, check if the entry in $(MSGDIR)/mail_include/run_MessageStore is correct. If there is no local problem contact the system and network administrators to determine the status of the email system.

3.4.9. Monitoring the Processing Status in the Database

Abstract:
The primary tools to monitor the data processing status are WorkFlow and RequestFlow, which show graphical representations of the IDCX.interval and IDCX.request tables. An alternative monitoring method is to directly check the contents of the database tables using SQL queries.

Additional information:
[IDC5.1.1] Database Schema,
[IDC5.1.3] Configuration of PIDC Databases.

Procedure:
The interactive monitoring tools offer a quick overview of the system status, while SQL queries will reveal more detailed information to trace specific issues across different processing applications or subsystems. Further, there are important database tables, e.g. msgdisc, which cannot be monitored with the existing interactive monitoring tools.

A good understanding of the IDC database schema, and especially the entity relationships between database tables, is very helpful to relate information from different tables and track processing information and the data flow through the system. Monitoring the database tables often helps to determine the origin of specific processing problems, since almost all IDC applications write status information to various database tables.

The automatic processing applications use, with only a few exceptions, the tables in the IDCX account. There are separate accounts SEL1, SEL2, SEL3 for the automatic bulletin results, LEB for interactive analysis, REB for the reviewed event bulletin, SEGMENT for the
segment archiving pipeline and MIGRATE for MigrateData. The MAP and STATIC accounts hold map and station configuration data and are only manually changed.

The most relevant tables for automatic processing are the interval, request, msgdisc and wfdisc tables, all in the IDCX account. They should be checked whenever there is an issue with data reception or automatic processing. More details are given in the procedures for individual tables.

3.4.10. IDCX.interval Table

Abstract:
IDCX.interval table plays the central role in the IDC data processing and archiving. It contains all available processing status information mapped to data time segments of various lengths.

Additional information:

Procedure:
- The IDCX.interval table holds information on the status of processing for each data interval and for each processing step. It is monitored using WorkFlow. Alternatively SQL queries can be used to, e.g. to find specific intervals, get all failed intervals or update the status of an interval.
- To find a specific interval:
  sqlwrap IDCXDB
  select * from interval where intvlid=<interval_id>;

- To find failed intervals for a given processing pipeline:
  select * from interval where class='<class>' and name='<name>' \
  and state like '%failed' and lddate> sysdate-\%N;

  where N is the amount of days behind the real time that guarantees that failed interval dates are included.
- To update the status of an interval:
  update interval set state='<new_state>' where \ 
  intvlid=<interval_id>;

  Note that this will not update the corresponding Tuxedo processing queue but only the database. Use WorkFlow to queue the interval for DACS processing as described in section 3.4.2.
- Get numbers of failures by category for a given time range:
  select class, name, state, count(*) from interval where \ 
  class='<class>' and name='<name>' and \ 
  time='<lower_limit>' and endtime='<upper_limit>' and \ 
  state like '%$failed' group by class, name, state;
3.4.11. IDCX.request Table
Abstract:
IDCX.request is the main table containing all IDC requests for external data.

Additional information:

Procedure:
• The IDCX.request table holds information on the status of data requests, which are created by the Retrieve Subsystem in order to get additional data from auxiliary stations. It is monitored by RequestFlow. Alternatively SQL queries can be used to, e.g., trace specific requests, get the number of requests with missing data or update the status of a request.

  • To find a specific request:
    sqlwrap IDCXDB
    select * from request where reqid=<request_id>;

  • Get numbers of running requests for each station and channel and for a given time range:
    select sta, chan, count(*) from request where state='running' \ and time >= <lower_limit> and endtime <= <upper_limit> \ group by sta, chan;

  • Get number of requests per state:
    select state, count(*) from request group by state;

  • To update the status of a request:
    update request set state='<new_state>' where reqid=<request_id>

    Note that WaveGet will queue any requests in status requested or retry if they are within its lookback time. However, manually updating requests to status queued will not automatically queue a Tuxedo message.

3.4.12. IDCX.msgdisc Table
Abstract:
IDCX.msgdisc table contains all validated incoming messages. It allows connection between msgid and the actual physical file.

Additional information:

Procedure:
• The IDCX.msgdisc table is an inventory of all incoming and outgoing email messages, which are received or sent by the Message Subsystem. The most important message types are request messages, data messages and subscription messages. The message table also holds information on the processing status of all messages. It is not monitored by any of the interactive monitoring tools but exclusively by manual database queries.
• To find a specific message, msgid can be easily found in the corresponding log file:
  sqlwrap IDCXDB
  select * from msgdisc where msgid=<message_id>;

• To find messages sent from a given email address (defined in the dispatch.par) or by a given user:
  select * from msgdisc where isrc='<email_address>';
  select * from msgdisc where userid in (select userid from datauser where username='<username>' and domain='<domain>');

• To update the status of a message (MessageReceive will re-process any messages in status RECEIVED):
  update msgdisc set status='<new_status>' where msgid=<message_id>;

• Get numbers of messages per status for the last 7 days:
  select status, count(*) from msgdisc where lddate > sysdate-7 group by status;

• If there are messages with a status RUNNING and there is an increasing number of messages in status RECEIVED a message-processing queue is blocked or unable to process incoming messages in a run-time pace. This typically occurs after a crash of an application, which left it in status RUNNING. Update the message to change the status from RUNNING to RECEIVED in order to re-process the message, or to FAILED in order to avoid re-processing.

• To find a request message that originates from a data request in the request table:
  select * from msgdisc where intidtype='reqid' and intid in (select reqid from request where array='<sta>' and start_time between '<start_time>' and '<start_time>'+1);

• To find a data response message for a given request message that was generated by the Retrieve Subsystem:
  select * from msgdisc where intidtype='wmreq_msgid' and intid=(select msgid from msgdisc where intidtype='reqid' and intid in (select reqid from request where array='<sta>' and start_time between '<start_time>' and '<start_time>'+1));

3.4.13. IDCX.wfdisc Table

Abstract:
IDCX.wfdisc contains all primary and auxiliary data.

Additional information:

The wfdisc table is an inventory of all received and parsed waveform files, which are stored in the file system. It is not directly monitored by any of the interactive monitoring tools. However, it is indirectly monitored by the automatic data monitors controlled by the DACS scheduler and by WaveAlert. WaveAlert compares the wfdisc and request tables and updates the request status depending on the amount of data received in the wfdisc table. The data
monitors queue new station processing intervals for primary and auxiliary stations if at least 80% of the data in the interval exists in the \texttt{wfdisc} table. For more detailed \texttt{wfdisc} monitoring use the following or similar database queries.

**Procedure:**

- To find data for a given station and channel within a given time range:
  ```
  select * from wfdisc where sta='<sta>' and chan='<chan>' \ 
  and time <= <upper_limit> and endtime >= <lower_limit>; 
  ```

- To find auxiliary data which was parsed from a given data message in the \texttt{msgdisc} table (i.e. a response to an automatic data request from the Retrieve Subsystem):
  ```
  select distinct w.* from wfdisc w, wftag t, msgdisc m \ 
  where w.wfid=t.wfid and t.tagid=m.msgid t.tagname='msgid' \ 
  and m.msgtype='DATA' and m.msgid=<message_id>; 
  ```

### 3.4.14. Other Relevant Tables

**Abstract:** This section lists examples for other tables that the Development LAN operator should monitor.

**Additional information:**

**Procedure:**

There is a very large number of possible queries to link information from different database tables in order to extract useful information, especially when diagnosing an unknown problem with data processing. Refer to the Database Schema document and the entity relationships included in that document to learn how entries in different tables can be related using their primary and foreign keys. The following list does not attempt to list all those possibilities, but lists simple examples for the most relevant database tables for routine monitoring and checking the system configuration.

- **IDCX.datauser** and **IDCX.subuser** tables identify authorized users for messages and subscriptions:
  ```
  select * from datauser where username=lower('<username>') \ 
  and msgtype='<message_type>'; 
  ```

- **IDCX.subuser** where username=upper('<USERNAME>.');

- **MigrateData** status for migration and purging:
  ```
  sqlwrap MIGRATEDB 
  select * from mig_date where procclass='<MIG|PURG>' and \ 
  proctype='<instance_name>'; 
  ```

- **IDCX.lastid** table:
  ```
  sqlwrap IDCXDB 
  select keyvalue from lastid where keyname='<name>id'; 
  ```

**STATIC** tables:

- Check **STATIC.affiliation** networks for a given station:
  ```
  sqlwrap STATICDB 
  select * from static.affiliation where sta='<sta>'; 
  ```
- Check currently configured channels for a given station:
  
  ```sql
  select * from static.sitechan where sta in \
  (select sta from affiliation where net='<station_name>') \
  and offdate=-1;
  ```

- Check currently configured instruments for a given station:
  
  ```sql
  select s.sta, s.chan, i.* from static.instrument, static.sensor \
  where s.inid=i.inid and s.endtime>9999999999 and s.sta in \
  (select sta from static.affiliation where net='<station_name>');
  ```

- Check stations in currently configured REQUEST network:
  
  ```sql
  select * from static.affiliation where net='REQUEST';
  ```
3.5. Maintenance Procedures

3.5.1. Installation of New SHI Stations

Abstract:
The installation of new stations is accomplished with a combination of database and configuration changes that depends on the type of station to be installed. The generic station installation CINs list the detailed changes to be made for each station type. The Perl application `staconf` is able to automate the installation for the most important station types. However, in most cases, stations get first installed in the Testbed LAN. Then it is easier to install the station in the Development LAN just by copying the relevant configuration files and database table entries. This section describes manual and automated installation as well as a procedure for installation by synchronization with the Testbed LAN.

Additional information:
[IDC6.2.4] Configuration of IDC Processing Data Files,
[CIN-01797] Installation of Primary Three-Component Stations,
[CIN-01806] Installation of Seismic Array Stations,
[CIN-01627] Installation of Auxiliary Stations,
[CIN-01567] Installation of Hydro-Acoustic Stations,
[CIN-01798] Installation of Infrasound Stations.

Procedure:
The station configuration data for stations to be installed or upgraded is maintained by the IMS division and is sent in the form of an IMS parameter file to the IDC. The IDC Processing Engineers receive the IMS parameter file to install the station on the Testbed and later in the Operations LAN. The Development LAN station configuration follows the Testbed configuration, and new stations are typically installed on the Development LAN after they have been installed on the Testbed. Contact the IMS division or the IDC Processing Engineers to obtain the IMS parameter file for the station to be installed.

In addition to the station configuration data proper, there is configuration data to define how the station is integrated into the existing system. Such information is for example the association of a station to specific CDTools instance and the names and addresses of recipients to which data for this station shall be forwarded. If `staconf` is used to install the station, much of this additional configuration data can be defined in an IDC parameter file, which is used by `staconf` in combination with the IMS parameter file to install the new station in the system.

There are three optional methods to install new stations on the Development LAN:

- Manual installation: Follow the generic installation CIN for the type of station to be installed. All configuration files will be sequentially modified in ClearCase and promoted to the runtime system.
- Automated installation: Obtain the IMS parameter file according to the established procedures (Station Transfer Document) and generate the IDC parameter file from an existing sample, e.g., by obtaining the IDC parameter file that has been used to install the station on the Testbed LAN. Start `staconf` with both parameter files. `staconf` will come up with an interactive menu to guide the user through the installation steps. `staconf` will write
all modified files, new files, database changes and a list of changes made to existing files to a local directory. All changes and new files can be checked and verified in this local directory before they are installed in ClearCase and promoted to the runtime system.

- Synchronisation with the configuration and database entries of the Testbed LAN: All configuration files to be changed are merged from the Testbed R3_tst branch to the devlan_view in ClearCase and promoted to the runtime system. All database changes are copied from the Testbed main database (fogo) to the Development LAN main database (idcdev). Non-trivial merges will be needed for system-specific parameter files, which differ between the Testbed and the Development LAN but also contain station configuration data. In some cases it will be more convenient to change the file manually rather than to merge it, due to a largely different content and structure between the two systems (e.g. shared.par).

Depending on the recent change history of the Testbed and Development LANs it may get complicated to sort out which changes are exactly needed to correctly merge a new station and which changes are due to other modifications. In this case it may be simpler to synchronise the entire configuration structure and static database tables between both systems. However, a remaining complication is that recent development changes and solutions to configuration problems that have not yet been promoted to the Testbed LAN must be preserved on the Development LAN.

For already existing stations their old configuration has to be completely disabled in the database tables and configuration files need to be changed manually. The above-mentioned CINs provide guidance in most cases.

3.5.2. Tuxedo Configuration on a New Server

Abstract:
Though it is not a regular procedure, occasionally Tuxedo needs to be configured on new machines, or a new Tuxedo release has to be installed.

Additional information:
[IDC6.5.2] Distributed Application Control System (DACS) Software User Manual,
[IDC7.3.1] Distributed Application Control System (DACS).

Procedure:
Prior to any work by the Development LAN operator, the system administrator for the machine in question has to install the Tuxedo distribution on that machine.

- Login as user auto on the machine where Tuxedo is to be configured.
- Insert into the rcusers/auto file a command to start the tlisten process each time the machine boots.
- Execute the script crTuxHost: to create the /var/tuxedo/IDC3_process subdirectories:
crTuxHost mode=process
- If the machine shall be configured as primary or backup Tuxedo master update the ubb file accordingly then resolve its symbolic parameters and load the resolved file into Tuxedo.
cd $(CMS_CONFIG)/system_specs
solvepar ubb_process.tmpl > ubb_process.resolved
tmloadcf –c /ubb_process.resolved  # checks the ubb file syntax
tmloadcf –y /ubb_process.resolved  # loads the ubb file

• For Tuxedo primary or backup masters create a Tuxedo transaction log file:
  tmadmin
crdl –b 1000 –z /var/tuxedo/IDC3_process/TLOGS/tlog
crlog –m TUXHOST1  # make sure to use the correct TUXHOST
    # as defined in the ubb file
quit

• For Tuxedo primary or backup queue hosts create the queue space:
  crQspace

• Then create the Tuxedo queues in the new queue space:
  crDacsQueues

  The script will create the standard set of queues which are used for automatic processing at the IDC. To build tailored pipelines with fewer or different queues modify the script to use customized names for the queue space and queues.

• Review the content of the local /var/tuxedo/IDC3_process/ subdirectories to verify that all required files have been created as expected. If yes test if Tuxedo processes can be started using either Tuxpad or the tmadmin command line interface to Tuxedo. A sample command to start and shut down only the Tuxedo sysadmin processes on the Tuxedo master host is:
  tmboot –M
tmshutdown –M

  Verify that the expected processes do successfully start, and check the Tuxedo ULOG file for any unexpected messages.

3.5.3. Synchronizing the Development LAN Configuration with the Testbed LAN

Abstract:
The following procedure shall be followed in case the Development LAN station configuration and static database account is to be synchronized with the current Testbed version. Another reason for its application is the wish to transfer the configuration of a new station from the Testbed LAN to the Development LAN.

According to the established development cycle at the IDC, software and configuration changes are first implemented and tested in the Development LAN, then promoted and operationally tested on the Testbed and finally installed in the Operations LAN. However, practice has shown that there are occasionally changes, which have not undergone the full cycle and were directly implemented on the Testbed. Additionally, all stations are first installed on the Testbed and then promoted to other LANs from there. The reasons for this practice are largely historic and are related to the fact that the various LANs are maintained by different organisational entities within the IDC. To avoid a long term de-synchronisation of the IDC LANs, configuration and database changes have to be promoted backwards to the Development LAN in intervals of several months. Development changes and solutions to
recent problems have to be preserved on the Development LAN during this synchronisation process.

**Additional information:**
[IDC5.1.1] Database Schema,

**Procedure:**
- A synchronisation of the Development LAN station configuration with the Testbed has two reasons. First the new stations, which have been installed on the Testbed, will be installed on the Development LAN. And all other recent configuration changes that have been made on the Testbed LAN will be detected and can be merged to the Development LAN. The benefit of keeping the configuration of both systems closely related to each other is that results of software tests on the Development LAN will be more predictive for the behaviour of the software on the Testbed and Operations LANs.

The synchronisation process consists of two parts: synchronisation of static database tables and synchronisation of configuration files.

**Synchronisation of database tables:**
- The following tables need to be synchronised:
  - STATIC.affiliation
  - STATIC.forbeamaux
  - STATIC.instrument
  - STATIC.network
  - STATIC.sensor
  - STATIC.site
  - STATIC.siteaux
  - STATIC.sitechan
  - STATIC.sitepoll
  - STATIC.site_address
  - STATIC.stanet
  - MAP.colordisc
  - MAP.mapcolor
  - MAP.mapdisc
  - MAP.mapover
  - MAP.mappoint
  - MAP.overlaydisc
  - IDCX.channname
  - IDCX.chan_groups
  - IDCX.datauser
  - IDCX.lastid
  - IDCX.subsuser
  - IDCX.timestamp
Notes for specific tables:

- Some of the above tables contain directory entries (dir column). The Development LAN specific directory paths need to be preserved. These tables are: STATIC.instrument, MAP.colordisc, MAP.mapdisc, MAP.overlaydisc.

- Some stations may need to be affiliated with different networks on the Development LAN than on the Testbed, due to tests with specific stations. For example the REQUEST network may differ in the affiliation table. Also the way auxiliary station data is requested in the Testbed LAN is very different from that process in the Dev LAN.

- If stations are tested on the Development LAN while they are not yet installed on the Testbed their entries in the STATIC tables have to be preserved (core tables are site, sensor, sitechan, instrument, affiliation).

- The chan_groups table is of lower priority on the Development LAN since it is used by the Archive application, which is not routinely operated. If it is operated for specific tests the archiving classes in the chan_groups table may need to be different than on the Testbed.

- The datauser and subsuser tables do not need to be strictly synchronised. However, entries for data providers are needed on the Development LAN in order to receive data messages (i.e. auxiliary data) from these providers. Note that the Testbed is an additional data provider to the Development LAN, thus the corresponding datauser entry needs to be preserved. All entries for developers or other people who need access to the Development LAN Message and Subscription Subsystems need to be preserved. Generally it is recommended to have a superset of the Testbed entries in the datauser and subsuser tables of the Development LAN.

- In the lastid table only the keyvalue entries for 'chanid' and 'inid' need to be synchronised. An alternative to obtaining these values from the Testbed lastid table is to query for the maximum chanid in sensor and sitechan tables, and inid in the sensor and instrument tables after those have been synchronised with the Testbed tables.

- The timestamp table does not need to be strictly synchronised. For each new primary station an entry with procclass='DLTRIGGER', procname='<station>' is needed. The time column is automatically updated when new data are received, and the entry is needed by tis_server in order to be able to queue new station processing intervals.

Queries for table synchronisation:

- Login to the idcdev database on the Development LAN:
  sqlwrap <STATICDB|IDCXDB|MAPDB>

- Check for entries, which exist only on the Development LAN:
  select * from <account>.<table> minus select * from \<account>.<table>@fogo;

- Check for entries, which exist only on the Testbed LAN:
  select * from <account>.<table>@fogo minus select * from \\
  <account>.<table>;

- Save entries to be preserved in a temporary table:
create table tmp_<table> as select * from table where \\
<clause defining entries to be saved>;

• Copy the Testbed table to the Development LAN table:
  truncate <table>;
  copy from centre/data@fogo insert into <table> using \\
  (select * from <account>.<table>);

• Insert the saved entries from the temporary table and drop the temporary table (This will 
  work only if no uniqueness constraints are violated; some entries may have to be modified 
  before inserting in this case):
  insert into <table> select * from tmp_<table>;
  drop table tmp_<table>;
  commit;

• Repeat the queries to check for differences between the Development LAN and Testbed 
  tables. Verify the results and correct the entries in the Development LAN table as needed.

• Repeat the procedure for other tables in the same account, then for the remaining database 
  accounts to be synchronised.

• Query modifications for tables with directory (dir) columns:
  For directory columns the Development specific substring of the directory path needs to 
  be preserved. This is done with the SQL command replace. Example queries are shown for 
  the STATIC .instrument table.
  
  o Check for instrument entries, which exist only on the Development LAN:
    select * from instrument minus select \n    inid, insname, instype, band, digital, samprate, ncalib, ncalper,\n    replace(dir,'/tst/software/shi','/dvl/software/shi') dir, dfile, \n    rsptype,lddate from static.instrument@fogo;

  o Check for instrument entries, which exist only on the Testbed LAN:
    select inid, insname, instype, band, digital, samprate, ncalib, \n    ncalper, replace(dir,'/tst/software/shi','/dvl/software/shi') \n    dir, dfile, rsptype,lddate from static.instrument@fogo \n    minus select * from instrument;

  o Copy the Testbed instrument table to the Development LAN:
    truncate instrument;
    copy from centre/data@fogo insert into instrument using (select \n    inid, insname, instype, band, digital, samprate, ncalib, ncalper,\n    replace(dir,'/tst/software/shi','/dvl/software/shi') dir, \n    dfile, rsptype,lddate from static.instrument@fogo); 

The remaining queries stay as in the standard case described above.

Synchronisation of configuration files:
• A synchronisation of configuration files between the Testbed and Development LAN is in 
  fact a three-way merge operation between the R3_tst_config_view, the
devlan_view (both in ClearCase) and the Development LAN runtime versions. The need to synchronise all three comes from the design of views and branches in ClearCase:

All files, which have been modified at least once in the devlan_view, have versions on the devlan branch in ClearCase. Files, which have never been changed in the devlan_view, have no devlan branch in ClearCase. In the latter case the devlan_view shows the latest version on the R3_tst branch (if existing) or the main branch (otherwise). As a result the devlan_view may show versions of files, which have been modified on the Testbed but have not been installed on the Development LAN.

However, those versions, which are installed on the Development LAN (i.e. in the runtime directories), should always be labeled with the DEV_LAN label. A DEV_LAN view is configured to only show versions that carry the DEV_LAN label. Thus the DEV_LAN view shows the actually installed versions of files for the Development LAN. Since there are more than 7000 files in the configuration directory structure a systematic approach is needed to keep all views, the DEV_LAN label and the runtime versions synchronised. A generic procedure is described below.

General recommendations:

- Due to the very large number of files in the IDC software configuration directory the output of the recursive diff commands below may get very long and hard to handle, even though only lists of files are created. It is recommended to run the diff commands separately for each configuration subdirectory (app_config, earth_specs, host_config, station_specs, system_specs) in order to keep the resulting file lists better maintainable. Most differences are typically expected in the app_config and system_specs subdirectories. If the list of different files is still too long, split the procedure further by going down to the next level of individual subdirectories where needed.

- If new files or directories are created or merged in ClearCase, a new version of the directory on the next higher level will be created. To correctly synchronise all changed directories and newly created files start merging at the highest directory level which is different and continue merging all subdirectory levels sequentially, such that individual files are merged last. In case of the DEV_LAN view this rule applies to setting the DEV_LAN label. If any runtime files are not visible in the DEV_LAN view it is likely that the DEV_LAN label was not correctly applied to the correct directory versions at some level starting from the config directory downwards.

Identifying configuration file differences:

- Login as cmss on any Dev LAN machine.

- Check if the DEV_LAN view correctly shows all runtime versions:

  ```
  ct setview DEV_LAN
diff -r /vobs/idc/config $(CMS_CONFIG) | egrep "^Bin|^Only|^diff"
  ```

- Correct any differences by applying the DEV_LAN label to the correct ClearCase versions (needs ClearCase permission to apply the label):

---

1 Note that this DEV_LAN view has capital letters – the difference to the devlan_view is significant.
cleartool mklabel -rep DEV_LAN /vobs/idc/config/
<subdirectory_path>/<filename>@@<version_identifier>

• If the DEV_LAN label was correct but the runtime version was not, promote the DEV_LAN version to the runtime system:
  cp -pf /vobs/idc/config/<subdirectory_path>/<filename> $(CMS_CONFIG)/<subdirectory_path>/<filename>

• Check for differences between the devlan_view and the DEV_LAN view to find all new file versions that have been modified on the Testbed and do not have a devlan branch in CleanCase:
  ct setview devlan_view
diff -r /vobs/idc/config /view/DEV_LAN/vobs/idc/config | \
egrep "^Bin|^Only|^diff"

• Check all different files to determine if they shall be installed on the Development LAN. Typically there will be a number of such files containing configuration data for new stations that have recently been installed on the Testbed. To install these files on the Development LAN apply the DEV_LAN label and promote the files to the runtime system:
  cleartool mklabel -rep DEV_LAN \
  /vobs/idc/config/<subdirectory_path>/<filename>
cp -pf /vobs/idc/config/<subdirectory_path>/<filename> $(CMS_CONFIG)/<subdirectory_path>/<filename>

• If these files need to be modified for the Development LAN, login as user with checkout/checkin permission for /vobs/idc/config (auto and cmss do not have this permission) and create a new version on the devlan branch in CleanCase, before labelling and promoting the file:
  ssh <username>@<Dev LAN machine>
  newgrp configcm
cd /vobs/idc/config/<subdirectory_path>
  ct co <filename>
  vi <filename>
  ct ci <filename>
  #mklabel and cp commands as above

• Check for differences between the devlan_view and the R3_tst_config_view to find all differences for files that have a devlan branch (i.e. which have been modified at least once on the Development LAN):
  ssh cmss@<Dev LAN machine>
  ct setview devlan_view
t startview R3_tst_config_view
diff -r /vobs/idc/config \
/view/R3_tst_config_view/vobs/idc/config |\
egrep "^Bin|^Only|^diff"

• Check all different files to determine if the Testbed versions shall be installed on the Development LAN. This is the core part of the synchronisation process, since merges of
files that have been modified on both the Testbed and the Development LAN may be non-trivial. The actual complications for such merges depend on the change history of the individual files on both LANs and on the time when the last synchronisation was done on the Development LAN. Practice has shown that, if the synchronisation is performed in intervals of 3 to 5 months the number of non-trivial file merges will typically be below ten. Files can be merged on the command line (see ClearCase manual) or using the interactive ClearCase interface xclearcase. The interactive method of merging in xclearcase is recommended since all differences to be merged are clearly visible and can be individually selected. ClearCase checkout/checkin permission is needed to perform file merges.

- Label and promote all merged files. Login as cmss to promote files to the runtime system:
  ```
cleartool mklabel -rep DEV_LAN \
/vobs/idc/config/<subdirectory_path>/<filename>
```

  ```
ssh cmss@<Dev LAN machine>
ct setview devlan_view
/vobs/idc/config/<subdirectory_path>/<filename>
```

  ```
  cp -pf /vobs/idc/config/<subdirectory_path>/<filename> \ 
  $(CMS_CONFIG)/<subdirectory_path>/<filename>
  ```

- At this point there should be no differences between the devlan_view, the DEV_LAN view and the runtime versions on the Development LAN (unless new versions are currently being checked in for files that have no devlan branch in ClearCase). Verify that no differences are left and correct if needed:
  ```
  ssh cmss@<Dev LAN machine>
  ct setview devlan_view
  diff -r /vobs/idc/config $(CMS_CONFIG) | grep "^Bin|^Only|^diff"
  diff -r /vobs/idc/config /view/DEV_LAN/vobs/idc/config | \ 
  grep "^Bin|^Only|^diff"
  ```

- Verify that the remaining different files between the devlan_view and the R3_tst_config_view are only those files, which shall not be installed on the Development LAN:
  ```
  diff -r /vobs/idc/config \ 
  /view/R3_tst_config_view/vobs/idc/config | \ 
  grep "^Bin|^Only|^diff"
  ```

- After the synchronisation of database tables and configuration files is complete, restart the DACS, CDTools and Message/Subscription Subsystem. Verify that all subsystems and applications successfully process data. Correct any configuration and database problems and check if they are due to omissions in the synchronisation process. Once the data provider(s) for newly installed stations start sending data verify that the data are successfully received and processed. Check the IDCX.affiliation table and add the stations to additional processing networks if needed. Check other station-specific database tables and configuration files in combination with application log files if data reception or processing problems are found for new stations.
3.6. Troubleshooting Procedures

3.6.1. Log File Location for Different Applications

The logging area for all SHI processing applications is 
${\text{lan}}$/logs/shi (=${\text{LOGDIR}}$). All automatic and interactive DACS applications and MigrateData write their log files into subdirectories under the daily jdate directories, which are created by the cronmklog script running from the crontab file for auto. The cron job automatically creates the daily log directories for the next day (i.e. tomorrow) and removes the log directories from 15 days behind the current time.

Some applications use log directories, which are separate from the daily log directories. Most of these applications have their own log file management, typically done by numbering log files, with the lowest number pointing to the newest log file and the oldest log file being deleted when a new log file is created. The CDTools applications, the Archive application, the Message Subsystem, StaPro and several interactive analysis applications use such separate log directories.

When searching specific log files in the daily jdate directories, note that the automatic processing applications write to the jdate directory corresponding to the timestamp of the data being processed (data time), while the controlling tuxshell DACS instances write to the runtime jdate directory (today). Due to this behaviour log files from different applications processing the same data interval may be located in different jdate directories.

3.6.2. Routine Checks and Monitoring

The list below describes regular maintenance checks and actions, which the Development LAN operator has to perform:

- Check if the jdate log directory and its subdirectories exist for today and the previous few days. A missing daily log directory usually leads to a halt of all DACS processing.

- Check if the $(LOGDIR)$ partition has sufficient free space for the next daily log files (typically 200MB). Under normal conditions the log files are automatically purged, such that the total used space in the log file partition does not grow. However, a full log file partition will typically lead to widespread problems in the entire processing system.

- To find the log entries for a particular processing interval, get the interval start or end time from Workflow, convert into epoch time using the h2e command, and search for the epoch time in the relevant application log file. If the epoch time is not found under the current jdate directory also search in adjacent jdate directories.

- To find error or failure messages in a given log file use the grep command:

```
cd $(LOGDIR)/<jdate>/<app_logdir>
egrep -i "error|failure|signal|segment" <logfilename>
```
#use other keywords if this yields too much output,
#e.g., ora- or gdi-
3.6.3. Interpreting Error Messages and Resolving Problems

Abstract:
The specific log and error messages vary for each application. Refer to the IDC documentation for the application or subsystem to find more application-specific information. Many of the Software Design Manuals explicitly list error messages and their specific significance.

Additional information:
[IDC6.2.1] Release 2 Operations and Maintenance Seismic, Hydroacoustic, and Infrasonic System,
[IDC6.5.x] IDC Software User Manuals,
[IDC7.x.y] IDC Software Design Documents,

Procedure:
Error messages may indicate configuration problems, e.g. if a parameter is not found or cannot be read. In this case check the application parameter file and the higher-level parameter files which are included in the application configuration, and correct the configuration.

Error messages may also indicate the unavailability of system resources, e.g. if the database connection is unavailable, directories or files cannot be accessed, the network connection is unavailable, etc. Such problems may be transient, i.e. they will vanish after a short time (e.g. if too many processes tried to obtain the same system resources at the same time), or they may be persistent. The operator has to check manually the availability of the relevant system resources. For example, if a file cannot be read check if the file exists and what permissions are set:

```bash
ls -l <directory_path>/<filename>
```

All automatic processing applications are either run by `auto` or by `cmss`, thus these two users need to have the relevant read/write/execute permissions. If a problem with accessing system resources persists, contact the system administrator to diagnose the specific system problem.

Also some error messages may indicate the inability of an application to process a particular set of data. In this case the problem can be data-related (e.g. corrupt data file) or software-related. To further determine the nature of the problem, try to reproduce it for different data and determine the specific features of the data that may trigger the problem. If the problem is clearly data-related, the problem should be logged and reported for further assessment. If the problem can be related to a software defect, report the software problem. If the problem can be related to specific features, which may routinely occur in regular data intervals, report an enhancement request. See section 2.6.5. for reporting problems.

In addition to the application log files the `syslog` files in the mass store may be helpful in certain cases. They are maintained by the system administrator and are not further described in this document.
3.6.4. Solving Common Problems

Abstract:
If a system problem is found there is a chance that the same problem has already occurred before and there is a known solution or workaround. A list of the most common known problems and actions to resolve them is given below. The solutions for many of these common problems are described in more detail in section 3.4. Check the procedures for the specific subsystem and for the relevant database tables.

Additional information: Sections “Solving Common Problems” in IDC Software User Manuals (SUMs)

Observation: All DACS processing has stopped

Procedure:
Check in Tuxpad and in the daily DACS logfiles if the scheduler is active. Check if there is an element (i.e. the scheduler table) in the schedule queue and initialise the Scheduler if there is no scheduler table.
- Check if daily log directories for today and recent days exist and create them if needed.
- Check available disk space in `$(LOGDIR)`, `$(SHI_DATADIR)`, `$(MSGDIR)` and `$(SHI_PRODUCTDIR)`. Delete files if needed to make space available. Check the cleanup `cron` jobs for auto and use them to clean old data and file products.
- Check dacx log files in daily log directories and ULOG files in `/var/tuxedo/IDC3_process` to find specific DACS or Tuxedo related problems.
- Clean the shared memory resources and restart the DACS. If this does not help re-create the tlog file.

Observation: A processing pipeline has stopped processing

Procedure:
- Check in Scheduler window if the pipeline is already unstalled and unstall if not.
- Check in Tuxpad if all tuxshell and corresponding TMQFORWARD processes are running and start them if needed.
- Check in Tuxpad if the data monitor for the pipeline (e.g. tis_server for station processing) is running and restart if needed.
- Check the log files for application and DACS processes in the pipeline. If no intervals are queued the log file of the data monitor may show the reason.
- Check the parameter files for the data monitor (ti*_server) and the tuxshell instances for the pipeline.

Observation: No continuous waveform data are received

Procedure:
- Check if all instances of the CDTools are running.
- Check if new frame files and waveform files are being written.
• Check the stationFile and dataconsumers files in the CDTools configuration directory to verify that the addresses for data providers (stations) and data recipients are correctly configured.

• Check if a network connection is available to the data provider and data recipient addresses:
  ping <IP_address>

• Contact the system administrator and network administrator to check the general system and network connectivity status.

• Contact data providers to check the sender status and data recipients to check if they are ready to receive continuous waveform data.

Observation: No message are received or sent by the Message Subsystem
Procedure:
• Check if all message and subscription processes are running.

• Check the status of messages in the IDCX.msgdisc table. If messages remain in status RUNNING for a long time and a growing number of messages is found in status RECEIVED update the status of the message in status RUNNING to either RECEIVED or FAILED to make the message queues available for other messages.

• Refer to sections 3.4.11. and 3.4.12. for more details.

Observation: Data requests remain in state running
Procedure:
• Check the status of Message Subsystem. Check the WaveAlert log file to verify that WaveAlert updates the status of requests.

• Check the station addresses in dispatch.par and correct them if needed. Also check entries in the IDCX.datauser table and consider cases when some station may have two separate email addresses to receive requests and send data.

• Contact the system administrator and network administrator to check the general system and network connectivity status.

• Check the email connectivity and sender status with the data providers.

Observation: No data requests are created by the automatic processing system
Procedure:
• Check the network processing and LP pipelines in WorkFlow and the WaveExpert log files.

• Check if station processing pipelines are processing current data. If no, or only a few stations are processed, the network processing pipelines may have too little data to generate events and will not request auxiliary data.

• Check the dispatch pipeline including the WaveGet data monitor, the tuxshell and TMQFORWARD process for dispatch, the dispatch script and MessageSend. Uninstall WaveGet if it is stalled.
Observation: Recall and post-analysis processing stopped

Procedure:

- Check if events are available in the LEB database account for the days that should be processed.

- Check if the LEB simulation cron job is active (see section 3.3.19.) and check its log file in the daily log directory for bulletin migration. Verify that the LEB.allocate_hour entries have been copied after bulletin migration was complete including all entries for the actions Del_Pass, Scan_Pass and RebDone.

- Check the data monitors, tuxshell and TMQFORWARD processes for the Recall and post-analysis pipelines in Tuxpad; uninstall pipelines; respectively start processes as needed.

- Check the corresponding DACS, tuxshell and application log files. DACS and tuxshell log files are found in the runtime jdate log directory, while application log files are found in the data time jdate log directory.

Observation: Some stations are not processed by certain applications or pipelines

Procedure:

- Check if data are received from these stations, and determine where the sequential data processing flow is interrupted to localise the problem.

- Check in the application parameter files which network affiliations are used.

- Check the affiliation table to determine if the stations are included in all relevant networks (i.e. those found to be used in the application parameter files). Add the stations to networks as needed.

- Check the station-specific processing configuration files and database tables. Compare with the Testbed configuration and database tables.

Observation: MigrateData intervals fail for purging cron jobs

Procedure:

- Check the relevant MigrateData log file to determine which differences have been detected between the main and archive databases.

- Correct the differences by copying the entries to the database instance where they are missing.

- Run the failed purge instance manually from the command line or wait for the cron job to run the next time.

- Verify that the run was successful, otherwise check if additional differences have been detected by MigrateData and correct them again.

Observation: Interactive applications do not open a window as expected

Procedure:

- Check if ssh correctly sets the DISPLAY variables including all three DISPLAY, DISPLAY1 and DISPLAY2. Make sure that ssh is invoked with the option -X (ssh -X) to properly forward displays.
Verify that the generic Development LAN user environment is set:

```bash
printenv | sort > file1
sort $(CMS_CONFIG)/system_specs/env/global.shenv > file2
diff file1 file2
```

**Observation:** Applications cannot be properly started

**Procedure:**

- Verify that the generic Development LAN user environment is set (see section 3.2.6.).
- Run the `mkCMSuser` script in the user home directory to generate the .dt environment and local .login and .cshrc files for the user. .cshrc will source the global.env system environment file, which sets the IMSPAR environment variable depending on CMS_CONFIG and CMS_MODE, such that the top-level system parameter files (shared.par and one of \analysis.par,public.par) will be available.

**Observation:** Other problems

**Procedure:**

- Use the appropriate monitoring procedures in section 3.4. to localise the problem in the system.
- Check the relevant log files (section 3.6.1.) and analyse the problem (section 3.6.3).
- Check the Software Design Documents and Software User Manuals in the IDC documentation to find additional information about the relevant applications and subsystems.
- Formally report the problem if no problem report exists for the problem found (section 3.6.5.).
- When a solution is found follow the applicable Development LAN change procedures in section 2.6.2. to implement and document the solution.
- Add the solution to this procedure or change other Development LAN procedures if this is useful to document the solution. See section 2.6.4. for changing Development LAN procedures.

### 3.6.5. Analysis and Logging of Unknown Problems

**Abstract:**
What to do if a problem occurs for which no procedure exists

**Additional information:** see [IDC6.5.2] chapter “Troubleshooting”, section “What can you do when … you do not know what is wrong?”

**Procedure:**

- If a data processing problem occurs and no solution or work around is known, determine the nature of the problem and collect specific information from the log files, database tables, data files, configuration files, system status, etc.
- If the problem is software-related check if there is an existing problem report for this type of problem. If not, submit a new problem report, attach all relevant information, log file sections, data base entries, data files, etc. in order to be able to reproduce the problem offline. If a problem report exists, attach to it all relevant new information. However,
avoid all unnecessary data in order to efficiently use the storage space of the problem database.

- If the problem is configuration-related try to reproduce it with alternative configurations, e.g. by changing individual application parameters. If a working configuration is found try to analyse what exactly caused the problem and determine if the working solution reliably resolves the problem. Assess if the same problem may also occur on other IDC LANs (Testbed, Operations) and submit a CIN to implement the solution if it is relevant for the other LANs.

- For problems leading to the total inability of an application, a subsystem or the entire processing system to process any data, determine if there were any recent changes made to the system, e.g. new software installed, configuration changed, station installed, infrastructure software upgraded. If the problem is related to a system change, roll back the change and verify if this resolves the problem. Check with other users of the Development LAN and with the system administrator if there was any such change.

If the problem is not related to a recent system change, it is likely related to missing or unavailable system resources. Typical cases are:

  - full log partition and/or non-existing daily log directory
  - overload of the GCI links and non-availability of email messages
  - file system not accessible
  - network connections unavailable or firewall blocking network connections
  - hardware failures or crashes
  - lack of swap space
  - process table full
  - power outages
  - non-availability of station data.

Try to find more evidence on the nature of the problem from the existing log files. Determine when the problem first occurred. Contact the system administrator to further diagnose the specific system problem. If a solution and/or workaround are found for a previously unknown problem, document the procedure that was used to solve the problem and add it to the existing Development LAN troubleshooting procedures (see section 3.6.).
TERMINOLOGY

**Glossary**

*ClearCase branch* A set of sequentially numbered versions within the version tree of a *ClearCase* object. All version trees have a main branch and can have additional branches with unique branch names.

*ClearCase label* A label that is applied to an individual version of a *ClearCase* object. Various labels are used for the Development LAN to define versions, which are part of a software baseline or patch as well as to define versions that are currently installed in the runtime system.

*ClearCase version tree* The set of all versions of a *ClearCase* object (i.e. a directory or file), which are stored in *ClearCase*. The version tree starts with version 0 on the main branch and can have additional branches, each holding sequentially numbered versions. At the IDC, the *ClearCase* version tree holds the history of files and directories on all IDC LANs as well as all development versions. A *ClearCase* version string consisting of branch names and the version number uniquely defines each version in the version tree.

*ClearCase view* A configured set of rules in *ClearCase* to select unique versions from all version trees of objects in a given VOB. Individual views are identified by view names. The views used for the Development LAN select versions based on their branch names or on specific labels.

*ClearCase VOB* A logical area in *ClearCase* to manage a set of *ClearCase* objects including all their versions. The IDC *ClearCase* VOBs contain high-level system directories including all their subdirectories, files and links. Each *ClearCase* VOB corresponds to a directory on the runtime system.

*cron job* *Cron* is a utility of the UNIX/Linux operating systems to automatically run individual commands, scripts or applications at configurable regular times. All IDC scripts and applications, which are not controlled by the DACS or by other scripts or applications, are controlled by cron. Individual *cron* jobs can be configured in the local *crontab* file for individual authorised users on each machine.

*data* Files written by the IDC processing system are considered data if they are reused by the system for further processing. Otherwise they are considered products. This categorisation is used in the Development LAN directory structure. Data and product files are not under *ClearCase* version control.

*Development LAN* The computer hardware and infrastructure used to integrate and test software and configuration changes as well as new software at the IDC before promoting the changes to the Testbed LAN for operational testing. Physically separate from the Operations and Testbed LANs and under less rigorous configuration control.
devlan branch  The devlan_view is configured to automatically create a new version on the devlan branch if a directory or file is modified and checked-in in the devlan_view. The latest sequential version on the devlan branch is always the latest version that has been created under the devlan_view. If the version tree of a ClearCase object does not have a devlan branch this ClearCase object has never been modified and checked-in in the devlan_view.

devlan_solaris_view  The devlan_solaris_view is exclusively used to build Linux compatible software on Solaris platforms for backwards compatibility reasons. It is configured to check in versions on the devlan_solaris branch. Since source code is uniformly maintained on the devlan branch for both Solaris and Linux platforms, the view will select the latest source code version on the devlan branch if this branch exists in the version tree of the ClearCase object. If no devlan branch exists for a particular object the view will use the same selection rules as the standard devlan_view. Only ClearCase objects which are installed via the Solaris build procedure in this view are supposed to have a devlan_solaris branch.

devlan_view  The devlan_view is the standard ClearCase view to be used to modify and check-in versions of ClearCase objects for the Development LAN. It is configured to select versions based on their branch names. It will select the latest version on the devlan branch if this branch exists in the version tree of the ClearCase object. It will select the latest version on the R3_tst branch if the devlan branch does not exist. If neither a devlan branch nor an R3_tst branch exists in a given version tree, the devlan_view will select the latest version of the ClearCase object on the main branch. This latter case occurs for files or directories that have never been modified either on the Development LAN or on the Testbed.

DEV_LAN label  The DEV_LAN label defines ClearCase versions, which are currently installed in the Development LAN runtime system. It can only be applied to a single unique version in the version tree of each ClearCase object. The DEV_LAN label is manually applied when a new version is promoted from ClearCase to the Development LAN runtime system.

DEV_LAN view  The DEV_LAN view is configured to select only versions, which have the DEV_LAN label applied. If the DEV_LAN labels are correctly set the DEV_LAN view will show exactly the same versions as the Development LAN runtime system directories.

Operations LAN  The computer hardware and infrastructure used for operational data processing at the IDC.

Pipeline  Well-defined processing sequence during automated processing, where the same data time interval is successively subjected to processing by several applications.
products  Files written by the IDC processing system are considered products if they are not further processed and are made available to (external) users. Otherwise they are considered data. This categorisation is used in the Development LAN directory structure. Data and product files are not under *ClearCase* version control.

runtime system  The directories and files, which are mounted on the machines of an IDC LAN and which are used to operate the IDC processing software. All directories and files in the runtime system, which are under version control, have a counter part in *ClearCase*. The Development LAN runtime versions correspond to *ClearCase* versions that have the DEV_LAN label applied. Data files, products and log files are not under version control and exist only on the runtime system.

Testbed LAN  The computer hardware and infrastructure used to test new stations and software changes in an operational environment before installation in the Operations LAN. A close copy but physically separate and independent from the Operations LAN.
### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATM</td>
<td>Atmospheric Transport Modelling</td>
</tr>
<tr>
<td>CCB</td>
<td>Configuration Control Board</td>
</tr>
<tr>
<td>CDS</td>
<td>Continuous Data Subsystem</td>
</tr>
<tr>
<td>CIN</td>
<td>Change Implementation Note</td>
</tr>
<tr>
<td>CSCI</td>
<td>Computer Software Configuration Item</td>
</tr>
<tr>
<td>CRP</td>
<td>Change Request Proposal</td>
</tr>
<tr>
<td>CTBTO</td>
<td>Comprehensive Nuclear-Test-Ban Treaty Organisation</td>
</tr>
<tr>
<td>DACS</td>
<td>Distributed Application Control Subsystem</td>
</tr>
<tr>
<td>DCR</td>
<td>Development LAN Change Request</td>
</tr>
<tr>
<td>Dev LAN</td>
<td>Development LAN</td>
</tr>
<tr>
<td>ECMWF</td>
<td>European Centre for Medium-Range Weather Forecasts</td>
</tr>
<tr>
<td>FOR</td>
<td>Field of Regard</td>
</tr>
<tr>
<td>IDC</td>
<td>International Data Centre</td>
</tr>
<tr>
<td>IMS</td>
<td>International Monitoring System</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>LEB</td>
<td>Late Event Bulletin</td>
</tr>
<tr>
<td>NCEP</td>
<td>U.S. National Centers for Environmental Prediction</td>
</tr>
<tr>
<td>NDC</td>
<td>National Data Centre</td>
</tr>
<tr>
<td>PSR</td>
<td>Possible Source Region</td>
</tr>
<tr>
<td>PTS</td>
<td>Provisional Technical Secretariat</td>
</tr>
<tr>
<td>REB</td>
<td>Reviewed Event Bulletin</td>
</tr>
<tr>
<td>RN</td>
<td>Radio-Nuclide technology</td>
</tr>
<tr>
<td>SEL</td>
<td>Standard Event List</td>
</tr>
<tr>
<td>SHI</td>
<td>Seismic, Hydro-acoustic and Infrasound technologies</td>
</tr>
<tr>
<td>SI</td>
<td>Software Integration unit</td>
</tr>
<tr>
<td>SRS</td>
<td>Standardized Source-Receptor Sensitivity</td>
</tr>
<tr>
<td>VOB</td>
<td>Versioned Object Base</td>
</tr>
</tbody>
</table>
REFERENCES

[ATM-NCEPDATA_V2.0] ATM NCEPDATA 2.0 Software User Guide.
[ATM-plot-FOR_2.0] ATM plot-FOR 2.0 Software User Guide.
[ATM-batchmodeATM_1.0-SIP] batchmodeATM 1.0 Software Installation Plan.
[ATM-SRSM_1.0] ATM SRSM-Model 1.0 Software User Guide.
[CIN-01797] Installation of Primary Three-Component Stations.
[CIN-01806] Installation of Seismic Array Stations.
[CIN-01627] Installation of Auxiliary Stations.
[CIN-01567] Installation of Hydro-Acoustic Stations.
[CIN-01798] Installation of Infrasound Stations.
[CIN-01462] Installation of new IMS Radionuclide Stations on the Testbed.
[IDC3.4.2] Formats and Protocols for Continuous Data CD-1.0, Revision 0.1.
[IDC3.4.3] Formats and Protocols for Continuous Data CD-1.1, Revision 0.3.
[IDC5.1.1] Database Schema, Revision 5.
[IDC5.1.3] Configuration of PIDC Databases, Revision 1.
[IDC5.2.1] IDC Processing of Seismic, Hydroacoustic, and Infrasonic Data, Revision 1.
[IDC5.2.2] IDC Processing of Radionuclide Data, Revision 3.
[IDC6.5.2] Distributed Application Control System (DACS) Software User Manual, Revision 0.1.

May-08
[IDC7.1.1] Detection and Feature Extraction (DFX) Scheme Files.
[IDC7.1.12] Station Processing (StaPro).
[IDC7.3.1] Distributed Application Control System (DACS).
[IDC7.4.2] Message Subsystem.
[IDC7.4.3] Retrieve Subsystem.
[IDC7.4.4] Subscription Subsystem.
[IDC7.5.1] Archiving Subsystem.