

LXD

Linux Cross Documenter

User Requirements Document

Issue: 1
Revision: 1

Reference: URD-LXD
Created: 27th March 2006
Last modified: 18th April 2006

Prepared by: Guillermo López Alejos

Abstract

The user requirements document states the functionalities and constraints that define the system.

Document Status Sheet

1. LXD Linux Cross Documenter. User Requirements Document			
2. Document Reference Number: URD-LXD			
3. Issue	4. Revision	5. Date	6. Reason for change
1	0	27-04-2006	
1	1	18-04-2006	First revision.

Table 1: Document Status Sheet

Document Change Record

Document Change Record		DCR No.	1
		Date	17-04-2006
		Originator	Guillermo López Alejos
		Approved By	Guillermo López Alejos
1. Doc. Title		User Requirements Document	
2. Doc. Ref. Num.		URD-LXD	
3. Doc. Issue / Rev. Num.		1 / 1	
4. Page	5. Paragraph	6. Reason for Change	
1	N.A.	Document status sheet was missing.	
1	1.2.1	<i>System</i> definition was missing.	
4	N.A.	Further description was required.	
5	N.A.	Further description was required.	
6	N.A.	Further description was required.	
8	3.2.2	User requirements simplified in one for clarity.	
9	3.2.3	User requirement regarding data model methodology needed clarification.	

Table 2: Document Change Record 1

Contents

Abstract	I
Document Status Sheet	I
Document Change Record	II
Table of contents	III
List of figures	IV
List of tables	V
1. Introduction	1
1.1. Purpose	1
1.2. Definitions, acronyms and abbreviations	1
1.2.1. Definitions	1
1.2.2. Acronyms	1
1.2.3. Abbreviations	1
1.3. References	2
1.4. Overview	2
2. General Description	3
2.1. General capabilities	3
2.2. General constraints	3
2.3. User characteristics	3
2.4. Operational environment	7
3. Specific Requirements	8
3.1. Capability requirements	8
3.2. Constraint requirements	8
3.2.1. Process constraints	8
3.2.2. User interface constraints	8
3.2.3. Standards constraints	9

List of Figures

1. Use cases for *LXD* 3

List of Tables

1.	Document Status Sheet	I
2.	Document Change Record 1	II
3.	Generate documentation tree use case description	4
4.	Synchronize documentation tree use case description	5
5.	Generate output documentation use case description	6

1. Introduction

This section will cope with the aim of both, the software and the document itself, definitions and references that will appear in the document, and a brief overview of the rest of the sections.

1.1. Purpose

The purpose of this document is to state the user requirements for *LXD*.

1.2. Definitions, acronyms and abbreviations

The aim of this section is to state all the definition of key terms, the acronyms and the abbreviations that appear along this document.

1.2.1. Definitions

Documentation tree Directory structure containing intermediate documents

Intermediate document Document that can be processed directly by the system in order to obtain the output documentation. It may be generated automatically from source code, written by hand by developers, or both

Output documentation Human readable documentation

Symbol source Resource that contains information about the symbols to process (i.e. source code or index file)

System Synonym of *LXD*. Set of programs that implement *LXD* capabilities

1.2.2. Acronyms

LXD Linux Cross Documenter

UR User Requirements

URD User Requirements Document

XML Extensible Markup Language

1.2.3. Abbreviations

CN Constraint Requirement

CP Capability Requirement

1.3. References

Not applicable.

1.4. Overview

This document is organized as follows:

- Section 2 describes general factors that affect *LXD* and its requirements. It includes a description of the different use cases.
- Section 3 contains a specific description of capabilities and constraints.

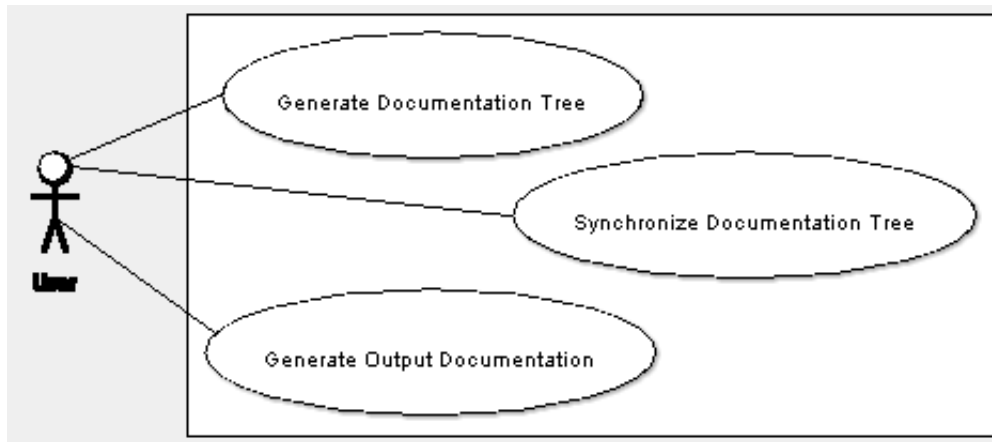


Figure 1: Use cases for *LXD*

2. General Description

This section provides an overview of the main aspects of *LXD*, such as capabilities, constraints, user characteristics and operational environment.

2.1. General capabilities

This section describes the process to be supported by *LXD* and its main capabilities. Use case techniques are applied.

Figure 1 shows the use cases for *LXD*. These use cases are described in their textual form in tables from 3 to 5.

2.2. General constraints

Since *LXD* is developed as a final year project by the author of this document, the first release will focus on the problem of getting documentation from the source code (i.e. document projects which are already implemented), without taking in account possible code generation capabilities.

2.3. User characteristics

Target *LXD* users are designers and code developers. They are intended to use it at implementation and maintenance phases.

Name	Generate documentation tree
Actors	User
Objective	Get the documentation tree generated. Documentation files inside the documentation tree are called intermediate documents
Pre-conditions	<ul style="list-style-type: none"> ■ The documentation tree does not exist at the given documentation tree path ■ The symbol source exists at the given symbol source path
Post-conditions	<ul style="list-style-type: none"> ■ The documentation tree was created at the given documentation tree path
Basic scenario	<p>This use case describes the situation in which the user uses the system to generate the documentation tree for the first time.</p> <ol style="list-style-type: none"> 1. The user specifies the path to the new documentation tree 2. The user specifies the path to the existing symbol source 3. <i>LXD</i> generates the documentation tree

Table 3: Generate documentation tree use case description

Name	Synchronize documentation tree
Actors	User
Objective	Get the documentation tree synchronized with the symbol source
Pre-conditions	<ul style="list-style-type: none"> ■ The documentation tree exists at the given documentation tree path ■ The symbol source exists at the given symbol source path
Post-conditions	<p>This use case describes the case in which the user uses the system to synchronize existing source code with an existing documentation tree.</p> <ul style="list-style-type: none"> ■ The documentation tree was updated applying the following policy <ul style="list-style-type: none"> ● Symbols present in the symbol source but not in the documentation tree, were created in the documentation tree ● Symbols present in the documentation tree but not in the symbol source, were marked in the documentation tree as <i>obsolete</i>. ● Different information related to the same symbol in the symbol source and the documentation tree, overwrote the documentation tree
Basic scenario	<ol style="list-style-type: none"> 1. The user specifies the path to the existing documentation tree 2. The user specifies the path to the existing symbol source 3. <i>LXD</i> synchronizes the the documentation tree with the symbol source

Table 4: Synchronize documentation tree use case description

Name	Generate output documentation
Actors	User
Objective	Get the output documentation generated from the documentation tree
Pre-conditions	<ul style="list-style-type: none"> ■ The documentation tree exists at the given documentation tree path ■ The output documentation path exists
Post-conditions	<ul style="list-style-type: none"> ■ The output documentation was written to the specified path ■ Existing output documentation was overwritten
Basic scenario	<p>This use case describes the case in which the user uses the system to generate output documentation for an existing documentation tree.</p> <ol style="list-style-type: none"> 1. The user specifies the path to the existing documentation tree 2. The user specifies the path for the output documentation files 3. <i>LXD</i> generates the output documentation in the given path

Table 5: Generate output documentation use case description

2.4. Operational environment

The operational environment for *LXD* is the framework of an open source project which code is written in C language.

3. Specific Requirements

This section states the user requirements for *LXD*. User requirements are identified as follows:

UR <Type>-<Number>

Where <Type> is either CP for *Capability Requirement* or CN for *Constraint Requirement*.

3.1. Capability requirements

Capability requirement are those requirements describing functions and operations needed by the user.

- UR CP-01** The system shall allow the user to generate the documentation tree.
- UR CP-02** The system shall allow the user to synchronize the symbol source with the documentation tree.
- UR CP-03** The system shall allow the user to generate output documentation.

3.2. Constraint requirements

This section specifies the requirements that restrict the software.

3.2.1. Process constraints

- UR CN-01** The system shall detect information mismatch between intermediate documents and symbol source.

3.2.2. User interface constraints

- UR CN-02** The system shall provide the user precise information about errors (syntax errors on intermediate documents or intermediate documents that are not valid within their specification) or decisions taken during processing.

3.2.3. Standards constraints

- UR CN-03** Each type of intermediate document shall have a syntax and semantic specification associated.
- UR CN-04** The specification of each intermediate document shall be accessible to the user.
- UR CN-05** Structure of intermediate documents shall comply with a procedural information model described by some methodology.