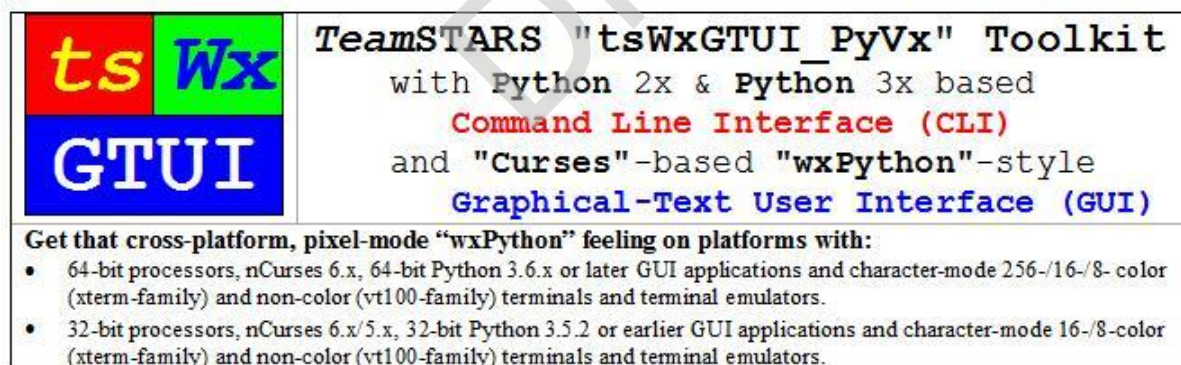


Release Notes

Vol. 8 - "tsWxGTUI_PyVx" Toolkit

Rev. 0.1.0 (Pre-Alpha)

Author(s): Richard S. Gordon



Author Copyrights & User Licenses for "tsWxGTUI_Py2x" & "tsWxGTUI_Py3x" Software & Documentation

- Copyright (c) 2007-2009 Frederick A. Kier & Richard S. Gordon, a.k.a. *TeamSTARS*. All rights reserved.
- Copyright (c) 2010-2017 Richard S. Gordon, a.k.a. *Software Gadgetry*. All rights reserved.
- GNU General Public License (GPL), Version 3, 29 June 2007
- GNU Free Documentation License (GFDL) 1.3, 3 November 2008

Third-Party Component Author Copyrights & User Licenses

- Attribution for third-party work directly or indirectly associated with the *TeamSTARS* "tsWxGTUI_PyVx" Toolkit are detailed in the "COPYRIGHT.txt", "LICENSE.txt" and "CREDITS.txt" files located in the directory named `./tsWxGTUI_PyVx_Repository/Documents`.

Draft

Contents

1	SCOPE (System Specification)	5
1.1	Identification (System Specification)	5
1.2	Purpose (System Specification)	8
1.3	Document Overview (System Specification)	10
2	DEFINITIONS, ABBREVIATIONS, AND ACRONYMS	23
3	SYSTEM REQUIREMENTS	55
3.1	Platform Configurations (Release Notes)	55
3.1.1	Currently Used Platforms (Release Notes)	57
3.1.2	Previously Used Platforms (Release Notes)	62
3.2	Network Configurations (Release Notes)	65
3.2.1	Stand Alone System Architecture	65
3.2.2	Stand Among System Architecture	69
3.3	Development & Test Platforms (Release Notes)	73
4	PACKAGE CONTENTS	75
4.1	Repository	76
4.1.1	Documents	85
4.1.2	ManPages	90
4.1.3	Notebooks	97
4.1.4	SourceDistributions	101
5	FIXED BUGS & ISSUES	121
5.1	Summary of Fixed Bugs & Issues	121
5.1.1	Fixed Bugs and Issues availble in Released Documents	122
5.1.2	Fixed Bugs and Issues available on Private Bugzilla Server	124
5.1.3	Fixed Bugs and Issues available on Public GitHub Repository	124
5.2	Details of Fixed Bugs & Issues	125
5.2.1	getOptions API	126
6	KNOWN BUGS & ISSUES	130
6.1	Design Issues	130
6.1.1	Xemacs Syntax Error	135
6.1.2	BuildPurpose AttributeError	136
6.1.3	Multiple GridSizer Mouse Click Events	136
6.1.4	Text & Password Entry Dialogs	137

6.2	Display Issues.....	137
6.3	Import Issues	138
6.4	Operational Issues	139
6.5	Installation Issues	139
6.6	Remote Access Issues	139
6.7	Troubleshooting Issues.....	140
6.7.1	Wing IDE Configuration	140
6.7.2	Debugging Instrumentation	146
6.7.3	Development Notes	147
6.8	User Input Issues	147

7 NEW FEATURES 149

7.1	User Interface Displays:	149
7.2	High-Level Widget API	150
7.3	GUI Events.....	153
7.4	Keyboard & Mouse Input.....	154

8 APPLICATION NOTES 163

8.1	README-GettingStarted	163
8.1.1	System Requirements	191
8.1.2	Python Programming Resources.....	202
8.1.3	Release Process.....	206
8.1.4	Release Repository	208
8.1.5	User Documentation	218
8.2	Installation Procedure.....	224
8.2.1	Developer Platform Procedure (Rough Draft).....	224
8.2.2	Operator Platform Procedure	225
8.3	User Interface Design	226
8.3.1	Command Line Interface	226
8.3.2	Graphical User Interface.....	227

8.4	Developer	228
9	PERFORMANCE TUNING	229

10	ACCEPTANCE TESTING	231
-----------	---------------------------	------------

11	COMPONENT STATUS (as of 2013/12/02)	233
-----------	--	------------

12	APPENDIXES	238
-----------	-------------------	------------

13	APPENDIX A - BASELINE HOST COMPUTER PLATFORMS	239
-----------	--	------------

13.1	Currently Used Platforms (Release Notes)	240
13.2	Previously Used Platforms (Release Notes).....	245

14	APPENDIX B - API RELATIONSHIP	249
-----------	--------------------------------------	------------

14.1	Comparison with “wxPython”	249
14.2	High, Low and Extended API Relationships	254

15	APPENDIX C - DELIVERABLES	277
-----------	----------------------------------	------------

Draft


1 SCOPE (System Specification)

Define the extent of the area or subject matter that something deals with or to which it is relevant.

- *Identification (System Specification)* (on page 5)
- *Purpose (System Specification)* (on page 8)
- *Document Overview (System Specification)* (on page 10)

1.1 Identification (System Specification)

This paragraph shall contain a full identification of the system and the software to which this document applies, including, as applicable, identification number(s), title(s), abbreviation(s), version number(s), and release number(s).

PRODUCT	IDENTIFICATION
Abbreviation	"tsWxGTUI"
Icon	
Name	<p>TeamSTARS "tsWxGTUI_PyVx" Toolkit</p> <p>Generic alias for the following Python language specific versions:</p> <ol style="list-style-type: none"> 1 TeamSTARS "tsWxGTUI_Py1x" Toolkit (reserved for legacy Python 1x; to be created by back-port from TeamSTARS "tsWxGTUI_Py2x") 2 TeamSTARS "tsWxGTUI_Py2x" Toolkit (for mature Python 2x; created as the Toolkit prototype; then upgraded as Python 2x evolved) 3 TeamSTARS "tsWxGTUI_Py3x" Toolkit (for evolving Python 3x; created as port from TeamSTARS "tsWxGTUI_Py2x"; then upgraded as Python 3x evolves) 4 TeamSTARS "tsWxGTUI_Py4x" Toolkit (reserved for future Python 4x; to be created by port from TeamSTARS "tsWxGTUI_Py3x"; then upgraded as Python 4x evolves)

Title	<i>Team</i> STARS "tsWxGTUI_PyVx" Toolkit with Python 2x & Python 3x based Command Line Interface (CLI) and "Curses"-based "wxPython"-style, Graphical-Text User Interface (GUI)
Identification Number	N/A
Version Number	0.1.0
Release Number	0.1.0 (pre-alpha)
Build Date	02/14/2017

Draft

This is free and open source software. The *TeamSTARS* "tsWxGTUI_PyVx" Toolkit and its third-party components are distributed in the hope that they will be useful, but WITHOUT ANY WARRANTY; WITHOUT EVEN THE IMPLIED WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Please note the following:

- 1 Each *TeamSTARS* "tsWxGTUI_PyVx" Toolkit distribution includes a root directory whose name ("tsWxGTUI") has a suffix "_PyVx" that reflects the associated Python language syntax version:

- a) **"tsWxGTUI_Py1x"** - *Reserved* for Root of first generation syntax files and subdirectories for Python 1.0.0-1.6.1.

There is currently no compelling need to justify the substantial effort to Backport from Python 2.x syntax, semantics and libraries.

There would be obsolete syntax and semantic issues to be resolved. For example, issues associated with the importing of modules and data.

There would be unimplemented library issues to be resolved. For example, Python 1.x supported only a Command Line Interface.

It would be necessary to Backport the Python 2.x standard curses library module in order to enable programmers to construct applications with a Text-based User Interface (TUI), which mimics the look and feel of a pixel-mode Graphical User Interface (GUI). Whereas a simple text-based interface that processes events from command-line interfaces operates sequentially, an advanced TUI may, like GUIs, use the entire screen area and accept mouse input. A TUI does not necessarily provide line-by-line output, although TUIs only use text, symbols and colors available on a given text environment.

- b) **"tsWxGTUI_Py2x"** - Root of second generation syntax files and subdirectories for **Python 2.0.0-2.7.13**.

Each Python 2.0.0-2.7.13 release typically includes the Python standard 32-bit curses-compatible nCurses 5.x library module.

The low level Application Programming Interface of the Python standard curses library module is a small, but usable, subset of the host platform's Text-based User Interface (TUI) capabilities.

The host platform's nCurses 5.9-/6.0-based curses library module is designed only for 32-bit processors (or for 64-bit processors operating in 32-bit compatibility mode) and consequently supports only 8-/16-Color palettes. For an overview of the evolution of BSD Curses and System V Release 4.0 (SVr4) to the latest nCurses release see the ***nCurses 6.0 Release Announcement*** (<http://invisible-island.net/ncurses/announce.html>).

- c) **"tsWxGTUI_Py3x"** - Root of third generation syntax files and subdirectories for **Python 3.0.0-3.5.2 and Python 3.6.0 or later**.

Each Python 3.0.0-3.5.2 release typically includes the Python standard 32-bit

1.2 Purpose (System Specification)

The TeamSTARS "tsWxGTUI_PyVx" Toolkit's cross-platform design facilitates the creation, enhancement, troubleshooting, maintenance, porting and support of:

- 1 Mission-critical equipment used by commercial, industrial, medical and military customers.
Such equipment can include a broad range of embedded computer systems which satisfy both the customer's application needs and the Toolkit's Platform Hardware and Software Requirements.
- 2 Application programs used for the local and remote supervisory control and data acquisition associated with automation, communication, control, diagnostic, instrumentation and simulation.
Such application software typically includes "operator-friendly" user interfaces.

The *TeamSTARS* "tsWxGTUI_PyVx" Toolkit includes the following software components:

- 1 **Release Distributions** - The Toolkit distribution in one or more Python language generation-specific form(s).
 - a) *TeamSTARS* "tsWxGTUI_Py1x" Toolkit for the first generation Python language 1.0.0-1.6.1 (reserved for future back-port from *TeamSTARS* "tsWxGTUI_Py2x" Toolkit)
 - b) *TeamSTARS* "tsWxGTUI_Py2x" Toolkit for the second generation Python language 2.0.0-2.7.13
 - c) *TeamSTARS* "tsWxGTUI_Py3x" Toolkit for the third generation Python language 3.0.0-3.5.2, 3.6.0 and later
 - d) *TeamSTARS* "tsWxGTUI_Py4x" Toolkit for the fourth generation Python language 4.0.0 (reserved for future port from *TeamSTARS* "tsWxGTUI_Py3x" Toolkit)
 - e) *TeamSTARS* "tsWxGTUI_PyVx" Toolkit reserved for distribution containing two or more of the above single generation Python language releases
 - f) *TeamSTARS* "tsWxGTUI_PyVx_SWIG" Toolkit reserved for the planned introduction of a Toolkit framework by which a third-party Simplified Wrapper and Interface Generator (SWIG) software development tool could connect programs written in C and C++ with a variety of high-level programming languages including common scripting languages such as Javascript, Perl, PHP, Python, Tcl and Ruby. Such a Toolkit addition could enable Python application software developers to create substitutes for the Python Standard Curses library module which have historically supported only a minimal subset of the C/C++ based nCurses Application Programming Interface.
- 2 **Toolkit Components** - Toolkit building-block components are general-purpose, re-usable and enable the application developer to focus on the application specific functionality and not waste effort re-inventing and re-implementing the functionality typical of Command Line and Graphical User Interfaces. Components include:
 - a) **tsToolkitCLI** - Python-based toolkit for development of applications featuring a Command Line Interface (CLI).

- b) **tsToolkitGUI** - Python and Curses-based toolkit for development of applications featuring a character-mode, test-based Graphical-style User Interface (GUI).

On newer platforms with 64-bit processors, nCurses-6.x and 64-bit GUI applications, the **tsToolkitGUI** provides that cross-platform, pixel-mode "wxPython" feeling on character-mode 8-/16-/256-color (xterm-family) and non-color (vt100-family) terminals and terminal emulators.

On legacy platforms with 32-bit processors, nCurses-6.x/5.x and 32-bit GUI applications, the **tsToolkitGUI** provides that cross-platform, pixel-mode "wxPython" feeling on character-mode 8-/16-color (xterm-family) and non-color (vt100-family) terminals and terminal emulators.

- 3 **Toolkit Applications** - Applications typically feature "user-friendly" Command Line and/or Graphical-style User Interfaces that can be controlled locally or remotely. Mission-critical equipment typically includes embedded computer systems which are customized and optimized for a specific use. Unlike their general-purpose desktop, laptop and workstation counterparts, they typically have limited, application-specific processing, memory, communication, input/output and file storage resources. Typical applications include:

- a) **Automation** - The use of various control systems for operating equipment such as machinery, processes in factories, telephone networks, steering and stabilization of ships, aircraft and other applications with minimal or reduced human intervention.
- b) **Communication** - The application of telecommunications technology for the transmission of data to, from, or between computers over dedicated or time shared hardware.
- c) **Control** - The application of one or more devices, to manage, command, direct or regulate the behavior of other device(s) or system(s). Industrial control systems, as used in industrial production, control equipment or machinery. In open loop control systems output is generated based only on inputs. In closed loop (feedback) control systems current output is taken into consideration and corrections are made based on feedback.
- d) **Diagnostic** - The application of technology to locate problems with software, hardware, or any combination thereof in a system, or a network of systems. Diagnostics typically provide guidance to the user to solve issues.
- e) **Instrumentation** - The application of technology for the measurement and control of process variables within a production or manufacturing area. An instrument is a device that measures a physical quantity such as flow, temperature, level, distance, angle, or pressure. Instruments may be as simple as direct reading thermometers or may be complex multi-variable process analyzers. Instruments are often part of a control system in refineries, factories, and vehicles.
- f) **Simulation** - The application of technology for the imitation of the operation of a real-world process or system over time. The act of simulating something first requires that a model be developed; this model represents the key characteristics or behaviors/functions of the selected physical or abstract system or process. The model represents the system itself, whereas the simulation represents the operation of the system over time.

1.3 Document Overview (System Specification)

This paragraph shall summarize the purpose and contents of this document and shall describe any security or privacy considerations associated with its use.

The Introduction is one of a set of reference document volumes for individuals installing, developing, maintaining, troubleshooting and using the *TeamSTARS* "tsWxGTUI_PyVx" Toolkit and application programs developed with the Toolkit. It and the other documents are described below.

VOL	TITLE	CONTENTS
0	Announcement	<p>This document alerts potential and existing users of the availability, capabilities, limitations and sources for the latest source code release and technical support.</p> <p>Included are the following topics:</p> <p>1 ANNOUNCEMENT</p> <p>a) What is it?</p> <p>Application Design</p> <p>b) How can you get it?</p> <p>Where to find it on GitHub?</p> <p>How to download it from Github as Clone (includes all releases) or ZIP file (includes only latest release)?</p> <p>c) What does it contain?</p> <p>Documents, ManPages, Notebooks, Source Distributions and release Manifest.</p> <p>d) How to get started?</p> <p>Read "GETTING_STARTED.txt". It is the first of a number of documents for Toolkit administrators and users. It identifies which hardware and software components you will need and any preparation required for Toolkit use.</p> <p>Read "README.txt". It is the first of a number of documents for Toolkit and User application software developers, troubleshooters and maintainers.</p>
1	Brochure	<p>From Wikipedia, the free encyclopedia:</p> <p>"A brochure (also referred to as a pamphlet) is a type of leaflet.</p> <p>Brochures are advertising pieces mainly used to introduce a company</p>

	<p>or organization, and inform about products and/or services to a target audience.</p> <p>Brochures are distributed by mail, handed personally or placed in brochure racks.</p> <p>The most common types of single-sheet brochures are the bi-fold (a single sheet printed on both sides and folded into halves) and the tri-fold (the same, but folded into thirds). A bi-fold brochure results in four panels (two panels on each side), while a tri-fold results in six panels (three panels on each side).</p> <p>Other folder arrangements are possible: the accordion or "Z-fold" method, the "C-fold" method, etc. Larger sheets, such as those with detailed maps or expansive photo spreads, are folded into four, five, or six panels. When two card fascia are affixed to the outer panels of the z-folded brochure, it is commonly known as a "Z-card".</p> <p>Booklet brochures are made of multiple sheets most often saddle stitched (stapled on the creased edge) or "perfect bound" like a paperback book, and result in eight panels or more.</p> <p>Brochures are often printed using four color process on thick gloss paper to give an initial impression of quality. Businesses may turn out small quantities of brochures on a computer printer or on a digital printer, but offset printing turns out higher quantities for less cost.</p> <p>Compared with a flyer or a handbill, a brochure usually uses higher-quality paper, more color, and is folded."</p> <p>Included are the following topics:</p> <p>1 INTRODUCTION</p> <p>a) About</p> <p>Platform Hardware and Software Requirements</p> <p>What is the too;kit designed to do?</p> <p>What is included in the release?</p> <p>How might you use the Toolkit?</p> <p>Where to get further information?</p> <p>b) System Block Diagrams</p> <p>Block Diagram</p> <p>Stand Alone System Architecture</p> <p>Stand Among System Architecture</p> <p>c) Usage Terms & Conditions</p> <p>2 SCREENSHOTS</p> <p>a) Latest XTERM & VT100 Desktops</p> <p>b) Earlier XTERM Terminal Emulator with 8-Color / 64-Color Pairs</p> <p>c) Earlier VT100 Terminal Emulator with 1-Color / 2-</p>
--	---

		Color Pairs
2	Introduction	<p>This document orients the reader to the goals and non-goals for the "tsWxGTUI_PyVx" Toolkit. Included are the following topics:</p> <ol style="list-style-type: none"> 1 SCOPE (System Specification) <ol style="list-style-type: none"> a) Identification (System Specification) b) Purpose (System Specification) c) Document Overview (System Specification) 2 SYSTEM OVERVIEW (tsWxGTUI Introduction) <ol style="list-style-type: none"> a) Purpose of System and Software b) General Nature of the System and Software c) History of System Development, Operation and Maintenance 3 OPERATOR INTERFACE <ol style="list-style-type: none"> a) Command Line Interface (CLI) b) Graphical User Interface (GUI) 4 APPLICATION PROGRAMMING INTERFACE <ol style="list-style-type: none"> a) Command Line Interface API b) Graphical User Interface API 5 TOOLS & UTILITIES <ol style="list-style-type: none"> a) tsLinesOfCodeProjectMetrics b) tsPlatformQuery c) tStripComments d) tsStripLineNumbers e) tsTreeCopy f) tsTreeTrimLines 6 DEFINITIONS, ABBREVIATIONS, AND ACRONYMS 7 REFERENCED DOCUMENTS <ol style="list-style-type: none"> a) Project Documents b) Release Distribution Documents c) External Documents 8 NOTES <ol style="list-style-type: none"> a) Operator Interface Technology 9 APPENDIXES <ol style="list-style-type: none"> a) Appendix A - Baseline Host Computer Platforms

		<ul style="list-style-type: none"> b) Appendix B - API Relationship c) Appendix C - Deliverables d) Appendix D - Accomplishments (Draft) e) Appendix E - Inherited, Field-Proven Computer Technology f) Appendix F - History of System Development. Operation, and Maintenance
3	Terms & Conditions	<p>This document alerts potential users and reminds existing users to the rules which the user must agree to abide by in order to use, modify and redistribute the "tsWxGTUI_PyVx" Toolkit and/or any of its components.</p> <p>Included are the following topics:</p> <p>1 TERMS & CONDITIONS</p> <ul style="list-style-type: none"> a) Copyright - Identifies the original author(s) of source code and documentation for building block libraries and application programs. b) License - Identifies the original author's rules for using, modifying and redistributing source code and documentation for building block libraries and application programs. c) Notices - Identifier placed on copies of the source code and documentation to inform the world of copyright ownership and the applicable license. d) Splash Screen Designer's Guide - Identifies the original author's personal guidelines for incorporating Copyright and License notices in Command Line Interface(s) and Graphical-style User Interface(s).
4	Software Development Plan	<p>This document specifies a developer's plans for conducting a software development effort. The term "software development" is meant to include new development, modification, reuse, re-engineering, maintenance, and all other activities resulting in software products.</p> <p>The plan provides the acquirer insight into, and a tool for monitoring, the processes to be followed for software development, the methods to be used, the approach to be followed for each activity, and project schedules, organization, and resources.</p> <p>Included are the following topics:</p> <p>1 SCOPE (System Specification)</p> <p>2 DEFINITIONS, ABBREVIATIONS, AND ACRONYMS</p> <p>3 OVERVIEW OF REQUIRED WORK</p> <ul style="list-style-type: none"> a) Purpose b) Requirements and Constraints

		<ul style="list-style-type: none"> c) Concept 4 PLANS FOR PERFORMING GENERAL SOFTWARE DEVELOPMENT ACTIVITIES <ul style="list-style-type: none"> a) Software Development Process b) General Plans for Software Development 5 PLANS FOR PERFORMING DETAILED SOFTWARE DEVELOPMENT ACTIVITIES <ul style="list-style-type: none"> a) Project Planning and Oversight b) Establishing A Software Development Environment c) System Requirements Analysis d) Software Design e) Software implementation And unit Test f) Unit integration And Testing g) CSCI Qualification Testing h) CSCI/HWCI Integration And Testing i) System Qualification Testing j) Preparing for Software Use k) Preparing for Software Transition l) Software Configuration Management m) Software Product Evaluation n) Software Quality Assurance o) Corrective Action p) Joint Technical and Management Reviews q) Other Software Development Activities 6 SCHEDULES AND ACTIVITY NETWORK 7 PROJECT ORGANIZATION AND RESOURCES <ul style="list-style-type: none"> a) Project organization b) Project Resources 8 NOTES 9 APPENDIXES 10 TO-DO-LIST <ul style="list-style-type: none"> a) New Features b) Modifications c) Troubleshooting
--	--	--

		<p>d) Validation/Regression Test</p> <p>11 SAMPLE SCREEN SHOTS</p>
5	System Specification	<p>This document specifies the required behavior of an engineering system. The documentation typically describes what is needed by the system user as well as requested properties of inputs and outputs (e.g. of the software system).</p> <p>Included are the following topics:</p> <p>1 SCOPE (System Specification)</p> <p>2 DEFINITIONS, ABBREVIATIONS, AND ACRONYMS</p> <p>3 REQUIREMENTS</p> <p>4 QUALIFICATION PROVISIONS</p> <p>a) Demonstration</p> <p>b) Test</p> <p>c) Analysis</p> <p>d) Inspection</p> <p>e) Special Qualification Methods</p> <p>5 REQUIREMENTS TRACEABILITY</p> <p>6 NOTES</p> <p>a) Partial List of Widget Toolkits</p> <p>b) Use Case(s)</p> <p>System Administrator</p> <p>Software Engineer</p> <p>System Operator</p> <p>7 APPENDIXES</p> <p>8 APPENDIX A - REQUIRED STATES AND MODES</p> <p>9 APPENDIX B - SYSTEM CAPABILITY REQUIREMENTS</p> <p>10 APPENDIX C - SYSTEM EXTERNAL INTERFACE REQUIREMENTS</p> <p>11 APPENDIX D - SYSTEM INTERNAL INTERFACE REQUIREMENTS</p> <p>12 APPENDIX E - SYSTEM INTERNAL DATA REQUIREMENTS</p> <p>13 APPENDIX F - ADAPTATION REQUIREMENTS</p> <p>14 APPENDIX G - SAFETY REQUIREMENTS</p> <p>15 APPENDIX H - SECURITY AND PRIVACY</p>

		<p>REQUIREMENTS</p> <p>16 APPENDIX I - SYSTEM ENVIRONMENT REQUIREMENTS</p> <p>17 APPENDIX J - COMPUTER RESOURCE REQUIREMENTS</p> <p>18 APPENDIX K - SYSTEM QUALITY FACTORS</p> <p>19 APPENDIX L - DESIGN AND CONSTRUCTION CONSTRAINTS</p> <p>20 APPENDIX M - PERSONNEL-RELATED REQUIREMENTS</p> <p>21 APPENDIX N - TRAINING-RELATED REQUIREMENTS</p> <p>22 APPENDIX O - LOGISTICS-RELATED REQUIREMENTS</p> <p>23 APPENDIX P - OTHER REQUIREMENTS</p> <p>24 APPENDIX Q - PACKAGING REQUIREMENTS</p> <p>25 APPENDIX R - PRECEDENCE AND CRITICALITY OF REQUIREMENTS</p>
6	Interface Requirements Specification	<p>This document specifies the requirements imposed on one or more systems, subsystems, Hardware Configuration Items (HWCI)s, Computer Software Configuration Items (CSCI)s, manual operations, or other system components to achieve one or more interfaces among these entities. An IRS can cover any number of interfaces.</p> <p>Included are the following topics:</p> <p>1 SCOPE (System Specification)</p> <p>2 DEFINITIONS, ABBREVIATIONS, AND ACRONYMS</p> <p>3 REQUIREMENTS</p> <p>a) Interface Identification and Diagrams</p> <p>b) (Project unique identifier of interface)</p> <p>c) Terminal Device Interface (TDI)</p> <p>d) Precedence and Criticality of Requirements</p> <p>4 QUALIFICATION PROVISIONS</p> <p>5 REQUIREMENTS TRACEABILITY</p> <p>6 NOTES</p> <p>7 APPENDIXES</p>
7	Software Requirements Specification	<p>This document specifies the requirements for a Computer Software Configuration Item (CSCI) and the methods to be used to ensure that each requirement has been met. Requirements pertaining to the CSCI's external interfaces may be presented in the SRS or in one or more Interface Requirements Specifications (IRSs).</p>

		<p>Included are the following topics:</p> <ol style="list-style-type: none"> 1 SCOPE (System Specification) 2 DEFINITIONS, ABBREVIATIONS, AND ACRONYMS 3 REQUIREMENTS <ol style="list-style-type: none"> a) Required States and Modes b) CSCI Capability Requirements c) CSCI External Interface Requirements d) CSCI Internal Interface Requirements e) CSCI Internal Data Requirements f) Adaptation Requirements g) Safety Requirements h) Security and Privacy Requirements i) CSCI Environment Requirements j) Computer Resource Requirements k) Software Quality Factors l) Design and Construction Constraints m) Personnel-related Requirements n) Training-related Requirements o) Logistics-related Requirements p) Other Requirements q) Packaging Requirements r) Precedence and Criticality of Requirements 4 QUALIFICATION PROVISIONS 5 REQUIREMENTS TRACEABILITY 6 NOTES <ol style="list-style-type: none"> a) TO-DO-LIST b) Command Line Interface Library c) Graphical Text User Interface Library 7 APPENDIXES <ol style="list-style-type: none"> a) Appendix A - Nature of System and Software 8 TECHNICAL IMPACT ANALYSIS 9 TEST REQUIREMENTS AND RESTRICTIONS 10 USE CASES 11 TRACEABILITY INFORMATION
--	--	--

		12 CLASS LIST BY CATEGORY
8	Release Notes	<p>This document is distributed with software products, often when the product is still in the development or test state (e.g., a beta release). For products that have already been in use by clients, the release note is a supplementary document that is delivered to the customer when a bug is fixed or an enhancement is made to the product.</p> <p>Included are the following topics:</p> <ul style="list-style-type: none"> 1 SCOPE (System Specification) 2 DEFINITIONS, ABBREVIATIONS, AND ACRONYMS 3 SYSTEM REQUIREMENTS <ul style="list-style-type: none"> a) Platform Configurations (Release Notes) b) Network Configurations (Release Notes) 4 PACKAGE CONTENTS <ul style="list-style-type: none"> a) Command Line Interface Library b) Graphical Text User Interface Library 5 FIXED BUGS & ISSUES 6 KNOWN BUGS & ISSUES 7 NEW FEATURES 8 APPLICATION NOTES <ul style="list-style-type: none"> a) Installation Procedure b) User Interface Design c) Developer 9 PERFORMANCE TUNING 10 ACCEPTANCE TESTING 11 COMPONENT STATUS 12 APPENDIXES <ul style="list-style-type: none"> a) Appendix A - Baseline Host Computer Platforms b) Appendix B - API Relationship c) Appendix C - Deliverables
9	Software Users Manual	<p>This document tells a hands-on software user (developer and operator) how to install and use the associated computer software (<i>TeamSTARS</i> "tsWxGTUI_PyVx" Toolkit). It may also cover a particular aspect of software operation, such as instructions for a particular position or task.</p> <p>Included are the following topics:</p> <ul style="list-style-type: none"> 1 SCOPE (System Specification) 2 SOFTWARE SUMMARY

		<ul style="list-style-type: none"> a) Software Application b) Software Inventory c) Software Environment d) Software Organization and Overview of Operation e) Security and Privacy f) Assistance and Problem Reporting <p>3 ACCESS TO THE SOFTWARE</p> <ul style="list-style-type: none"> a) First-time User of the Software b) Initiating a Session c) Stopping and Suspending Work <p>4 PROCESSING REFERENCE GUIDE</p> <ul style="list-style-type: none"> a) Capabilities b) Conventions c) Processing Procedures d) Related Processing e) Data Backup f) Recovery from Errors, Malfunctions, and Emergencies g) Messages h) Quick Reference Guide <p>5 NOTES</p> <ul style="list-style-type: none"> a) Use Case(s) b) Baseline Toolkit Development Platforms <p>6 DEFINITIONS, ABBREVIATIONS, AND ACRONYMS</p> <p>7 APPENDIXES</p> <p>8 APPENDIX A - BASELINE HOST COMPUTER PLATFORMS</p> <p>9 APPENDIX B - API RELATIONSHIP</p> <p>10 APPENDIX C - DELIVERABLES</p> <p>11 APPENDIX D - LOG FILES</p>
10	Appendixes	<p>1 Introduction</p> <ul style="list-style-type: none"> a) Appendix A - Baseline Host Computer Platforms b) Appendix B - API Relationship c) Appendix C - Deliverables d) Appendix D - Accomplishments (Draft)

		<ul style="list-style-type: none"> e) Appendix E - Inherited, Field-Proven Computer Technology f) Appendix F - History of System Development. Operation, and Maintenance
		2 Software Development Plan
		3 System Specification
		<ul style="list-style-type: none"> a) APPENDIX A - REQUIRED STATES AND MODES b) APPENDIX B - SYSTEM CAPABILITY REQUIREMENTS c) APPENDIX C - SYSTEM EXTERNAL INTERFACE REQUIREMENTS d) APPENDIX D - SYSTEM INTERNAL INTERFACE REQUIREMENTS e) APPENDIX E - SYSTEM INTERNAL DATA REQUIREMENTS f) APPENDIX F - ADAPTATION REQUIREMENTS g) APPENDIX G - SAFETY REQUIREMENTS h) APPENDIX H - SECURITY AND PRIVACY REQUIREMENTS i) APPENDIX I - SYSTEM ENVIRONMENT REQUIREMENTS j) APPENDIX J - COMPUTER RESOURCE REQUIREMENTS k) APPENDIX K - SYSTEM QUALITY FACTORS l) APPENDIX L - DESIGN AND CONSTRUCTION CONSTRAINTS m) APPENDIX M - PERSONNEL-RELATED REQUIREMENTS n) APPENDIX N - TRAINING-RELATED REQUIREMENTS o) APPENDIX O - LOGISTICS-RELATED REQUIREMENTS p) APPENDIX P - OTHER REQUIREMENTS q) APPENDIX Q - PACKAGING REQUIREMENTS r) APPENDIX R - PRECEDENCE AND CRITICALITY OF REQUIREMENTS
		4 Interface Requirements Specification
		5 Software Requirements Specification

		<ul style="list-style-type: none"> a) APPENDIX A - Nature of System and Software <p>6 Release Notes</p> <ul style="list-style-type: none"> a) APPENDIX A - Baseline Host Computer Platforms b) APPENDIX B - API Relationship c) APPENDIX C - Deliverables <p>7 Software Users Manual</p>
11	Dictionary	A reference document that contains words, phrases, abbreviations and acronyms that are listed in alphabetical order with information about their meanings in the context of the <i>TeamSTARS</i> "tsWxGTUI_PyVx" Toolkit software.
12	To-Do	A list of planned development for the code.
13	Use Cases	<p>A list of steps, typically defining interactions between a role (known in Unified Modeling Language (UML) as an "actor") and a system, to achieve a goal. The actor can be a human, an external system, or time.</p> <p>1 Computer User Use Case(s)</p> <ul style="list-style-type: none"> a) Role-Specific b) System-Specific c) Application-Specific <p>2 Computer Source Code Use Case(s)</p> <ul style="list-style-type: none"> a) Comparison with "wxPython" b) High, Low and Extended API Relationships

14	Operator Documents	<p>This document provides hands-on system administrators and equipment operators with information about the <i>TeamSTARS</i> "tsWxGTUI_PyVx" Toolkit.</p> <p>Included are reference copies of the following topic documents:</p> <ol style="list-style-type: none"> 1 "AUTHORS.txt" 2 "BUGS.txt" 3 "CHANGE_LOG.txt" 4 "CONFIGURE.txt" 5 "COPYING.txt" 6 "COPYRIGHT.txt" 7 "CREDITS.txt" 8 "DEMO.txt" 9 "FAQ.txt" 10 "GETTING_STARTED.txt" 11 "INSTALL.txt" 12 "LICENSE.txt" 13 "NEWS.txt" 14 "NOTICES.txt" 15 "OPERATE.txt" 16 "README.txt" 17 "README1-Introduction.txt" 18 "README2-Repository.txt" 19 "README3-Documents.txt" 20 "README4-ManPages.txt" 21 "README5-Notebooks.txt" 22 "README6-SourceDistributions.txt" 23 "README7-DeveloperSandboxes.txt" 24 "README8-SitePackages.txt" 25 "README9-KeyboardMouseInput.txt" 26 "THANKS.txt" 27 "TO-DO.txt" 28 "TROUBLESHOOT.txt"
----	--------------------	---

2 DEFINITIONS, ABBREVIATIONS, AND ACRONYMS

Trademarks and copyrights (http://en.wikipedia.org/wiki/Wikipedia:About#Trademarks_and_copyrights)

Wikipedia is a registered trademark of the not-for-profit *Wikimedia Foundation*, which has created a family of free-content projects that are built by user contributions.

Most of Wikipedia's text and many of its images are dual-licensed under the *Creative Commons Attribution-Sharealike 3.0 Unported License* (CC-BY-SA) and the *GNU Free Documentation License* (GFDL) (unversioned, with no invariant sections, front-cover texts, or back-cover texts). Some text has been imported only under CC-BY-SA and CC-BY-SA-compatible license and cannot be reused under GFDL; such text is identified either on the page footer, in the page history or on the discussion page of the article that utilizes the text. Every image has a description page which indicates the license under which it is released or, if it is non-free, the rationale under which it is used.

Contributions remain the property of their creators, while the CC-BY-SA and GFDL licenses ensure the content is freely distributable and reproducible. (See the **copyright notice** (<http://en.wikipedia.org/wiki/Wikipedia:Copyrights>) and the **content disclaimer** (<http://en.wikipedia.org/wiki/Wikipedia:Disclaimers>) for more information.)

The following terms are used throughout this document:

TERM	DEFINITION
API	An application programming interface (API) is a source code based specification intended to be used as an interface by software components to communicate with each other. An API may include specifications for routines, data structures, object classes, and variables.
Author-it®	<p>Excerpts From Wikipedia, the free encyclopedia:</p> <p>"Author-it® is a help authoring tool and content management system for creating, maintaining, and distributing single-sourced content.</p> <p>Author-it® can produce documentation in the following formats:</p> <ul style="list-style-type: none"> ▪ RTF, PDF, or Microsoft Word format for printed documentation ▪ Microsoft WinHelp (Windows Help) ▪ Microsoft HTML Help ▪ JavaHelp ▪ Oracle Help for Java ▪ Web and browser based help ▪ XML ▪ DITA

TERM	DEFINITION
	<p>Author-it® stores all information as objects in a central database, called a library. Object types include books, topics, file objects, hyperlinks, styles, glossaries, tables of contents, indexes, and publishing profiles. The library supports multiple users. Author-it® Base User module includes importing, authoring, and publishing capability. Further modules that can be licensed...."</p> <p>Product Information & Platform Constraints:</p> <ul style="list-style-type: none"> ▪ Author-it® is a product of the Author-it® Software Corporation, Ltd. ▪ Any chapter, section or paragraph component authored for a hierarchical location in one document (document 1 at a.b.c) may be re-used by reference at any other hierarchical location in any other document (document 2 at d.e, document 3 at f.g.h.i.j.k). Author-it® automatically re-numbers new references as appropriate for their new hierarchical location. ▪ Author-it® 4.5 and 5.0 are Workgroup Editions. They use the Microsoft JET Database Engine and are compatible with Windows XP Professional, Windows Vista and Windows 7 Professional. ▪ Author-it® 5.5 (trial) is the last stand-alone Workgroup Edition to use the Microsoft JET Database Engine. It is compatible with Windows XP Professional, Windows Vista, Windows 7 Professional and Windows 8 Professional. A Microsoft JET Database can be exported to now-obsolete versions of either the Microsoft SQL Server 2008 R2 or the free Microsoft SQL 2008 R2 Express Database. Since my Author-it 5 support subscription ended in 2007, I cannot obtain official Author-it confirmation of the following: (1) According to Microsoft, the Microsoft JET Database is not available on 32-/64-bit Microsoft Windows 10. (2) The now-obsolete Author-it 4.5 and 5 editions of Author-it® do not appear to work with a post 2008 release of Microsoft SQL and with an Office 365 (post XP/2003) version of Microsoft Word. ▪ Author-it® iCloud Professional and Enterprise Editions use either the Microsoft SQL Server or the free Microsoft SQL Express Database engine.
Automation	The use of various control systems for operating equipment such as machinery, processes in factories, telephone networks, steering and stabilization of ships, aircraft and other applications with minimal or reduced human intervention.
Berkley Software Distribution License	<p>From Wikipedia, the free encyclopedia</p> <p>"* * *. BSD licenses are a family of permissive free software licenses, imposing minimal restrictions on the redistribution of covered software. This is in contrast to copyleft licenses, which have reciprocity share-alike requirements. The original BSD license was used for its namesake, the Berkeley Software Distribution (BSD), a Unix-like operating system. The original version has since been revised and its descendants are more properly termed modified BSD licenses.</p> <p>Two variants of the license, the New BSD License/Modified BSD License (3-clause),[1] and the Simplified BSD License/FreeBSD License (2-clause)[2] have been verified as GPL-compatible free software licenses by the Free Software Foundation, and have been vetted as open source licenses by the Open Source Initiative,[3] while the original, 4-clause license has not been accepted as an open source license and, although the original is considered to be a free software license by the FSF, the FSF does not consider it to be compatible with the GPL due to the advertising clause.[4]"</p>
Building Blocks	A Building Block is a basic unit from which something of greater complexity is built up. For example, an application or operating system program may be constructed from one or more data structure definitions, functions, methods, classes, subroutines, threads, processes or building block libraries.
CLI	<p>Acronym for Command Line Interface.</p> <p>From Wikipedia, the free encyclopedia:</p>

TERM	DEFINITION
	A command-line interface (CLI) is an interface or dialog between the user and a program, or between two programs, where a line of text (a command line) is passed between the two.
CMAKE	<p>Excerpt From: https://cmake.org (https://cmake.org)</p> <p>"CMake is an open-source, cross-platform family of tools designed to build, test and package software. CMake is used to control the software compilation process using simple platform and compiler independent configuration files, and generate native makefiles and workspaces that can be used in the compiler environment of your choice. The suite of CMake tools were created by Kitware in response to the need for a powerful, cross-platform build environment for open-source projects such as ITK and VTK.</p> <p>CMake is part of Kitware's collection of commercially supported open-source platforms for software development."</p>
Communication	The application of telecommunications technology for the transmission of data to, from, or between computers over dedicated or time shared hardware.
Control	The application of one or more devices, to manage, command, direct or regulate the behavior of other device(s) or system(s). Industrial control systems, as used in industrial production, control equipment or machinery. In open loop control systems output is generated based only on inputs. In closed loop (feedback) control systems current output is taken into consideration and corrections are made based on feedback.
Curses	<p>From Wikipedia, the free encyclopedia:</p> <p>"curses is a terminal control library for Unix-like systems, enabling the construction of text user interface (TUI) applications.name is a pun on the term "cursor optimization". It is a library of functions that manage an application's display on character-cell terminals (e.g., VT100).[1]</p> <p>Overview</p> <p>The curses API is described in several places.[2] Most implementations of curses use a database that can describe the capabilities of thousands of different terminals. There are a few implementations, such as PDCurses, which use specialized device drivers rather than a terminal database. Most implementations use terminfo; some use termcap. Curses has the advantage of back-portability to character-cell terminals and simplicity. For an application that does not require bit-mapped graphics or multiple fonts, an interface implementation using curses will usually be much simpler and faster than one using an X toolkit.</p> <p>Using curses, programmers are able to write text-based applications without writing directly for any specific terminal type. The curses library on the executing system sends the correct control characters based on the terminal type. It provides an abstraction of one or more windows that maps onto the terminal screen. Each window is represented by a character matrix. The programmer sets up each window to look as they want the display to look, and then tells the curses package to update the screen. The library determines a minimal set of changes needed to update the display and then executes these using the terminal's specific capabilities and control sequences.</p> <p>In short, this means that the programmer simply creates a character matrix of how the screen should look and lets curses handle the work."</p>

TERM	DEFINITION
Cygwin	<p>From Wikipedia, the free encyclopedia:</p> <p>"Cygwin[2] is a Unix-like environment and command-line interface for Microsoft Windows. Cygwin provides native integration of Windows-based applications, data, and other system resources with applications, software tools, and data of the Unix-like environment. Thus it is possible to launch Windows applications from the Cygwin environment, as well as to use Cygwin tools and applications within the Windows operating context.</p> <p>Cygwin consists of two parts: a dynamic-link library (DLL) as an API compatibility layer providing a substantial part of the POSIX API functionality, and an extensive collection of software tools and applications that provide a Unix-like look and feel.</p> <p>Cygwin was originally developed by Cygnus Solutions, which was later acquired by Red Hat. It is free and open source software, released under the GNU General Public License version 3. Today it is maintained by employees of Red Hat, NetApp and many other volunteers."</p> <p>See also:</p> <ul style="list-style-type: none"> ▪ Cooperative Linux ▪ Cygwin/X (X11 for Cygwin) ▪ GnuWin32 ▪ Interix ▪ MinGW (Minimalist GNU for Windows) ▪ mintty (Cygwin terminal) ▪ UWIN
Darwin	<p>From Wikipedia, the free encyclopedia:</p> <p>"Darwin is an open source Unix-like computer operating system released by Apple Inc. in 2000. It is composed of code developed by Apple, as well as code derived from NeXTSTEP, BSD, and other free software projects.</p> <p>Darwin forms the core set of components upon which OS X and iOS are based. It is mostly POSIX compatible, but has never, by itself, been certified as being compatible with any version of POSIX. (OS X, since Leopard, has been certified as compatible with the Single UNIX Specification version 3 (SUSv3).[2][3][4])</p> <p>HISTORY</p> <p>Darwin's heritage began with NeXT's NeXTSTEP operating system (later known as OpenStep), first released in 1989. After Apple bought NeXT in 1997, it announced it would base its next operating system on OpenStep. This was developed into Rhapsody in 1997, Mac OS X Server 1.0 in 1999, Mac OS X Public Beta in 2000, and Mac OS X 10.0 in 2001. In 2000, the core operating system components of Mac OS X were released as open-source software under the Apple Public Source License (APSL) as Darwin; the higher-level components, such as the Cocoa and Carbon frameworks, remained closed-source.</p> <p>Up to Darwin 8.0.1, Apple released a binary installer (as an ISO image) after each major Mac OS X release that allowed one to install Darwin on PowerPC and Intel x86 computers as a standalone operating system. Minor updates were released as packages that were installed separately. Darwin is now only available as source code,[5] except for the ARM variant, which has not been released in any form separately from iOS. However, the older versions of Darwin are still available in binary form,[6] and a hobbyist developer winocm took the official Darwin source code and ported it to ARM.[7]"</p>

TERM	DEFINITION
Debian	<p>From Wikipedia, the free encyclopedia</p> <p>"Debian (/ˈdɛbiən/) is an operating system composed primarily of free and open-source software, most of which is under the GNU General Public License, and developed by a group of individuals known as the Debian project. Debian is one of the most popular Linux distributions for personal computers and network servers, and has been used as a base for several other Linux distributions.</p> <p>Debian was first announced in 1993 by Ian Murdock, and the first stable release was made in 1996. The development is carried out over the Internet by a team of volunteers guided by a project leader and three foundational documents. New distributions are updated continually, and the next candidate is released after a time-based freeze.</p> <p>As one of the earliest Linux distributions, it was envisioned that Debian was to be developed openly in the spirit of Linux and GNU. This vision drew the attention and support of the Free Software Foundation, which sponsored the project for the first part of its life."</p>
Developer Sandbox	<p>Excerpt From Wikipedia, the free encyclopedia:</p> <p>"A sandbox is a testing environment that isolates untested code changes and outright experimentation from the production environment or repository, in the context of software development including Web development and revision control. Sandboxing protects "live" servers and their data, vetted source code distributions, and other collections of code, data and/or content, proprietary or public, from changes that could be damaging (regardless of the intent of the author of those changes) to a mission critical system or which could simply be difficult to revert. Sandboxes replicate at least the minimal functionality needed to accurately test the programs or other code under development (e.g. usage of the same environment variables as, or access to an identical database to that used by, the stable prior implementation intended to be modified; there are many other possibilities, as the specific functionality needs vary widely with the nature of the code and the application[s] for which it is intended.)</p> <p>The concept of the sandbox (sometimes also called a working directory, a test server or development server) is typically built into revision control software such as CVS and Subversion (SVN), in which developers "check out" a copy of the source code tree, or a branch thereof, to examine and work on. Only after the developer has (hopefully) fully tested the code changes in their own sandbox should the changes be checked back into and merged with the repository and thereby made available to other developers or end users of the software.[1]</p> <p>By further analogy, the term "sandbox" can also be applied in computing and networking to other temporary or indefinite isolation areas, such as security sandboxes and search engine sandboxes (both of which have highly specific meanings), that prevent incoming data from affecting a "live" system (or aspects thereof) unless/until defined requirements or criteria have been met."</p>
Diagnostic	<p>The application of technology to locate problems with software, hardware, or any combination thereof in a system, or a network of systems. Diagnostics typically provide guidance to the user to solve issues.</p>

TERM	DEFINITION
Docstring	<p>Excerpt from http://www.python.org/dev/peps/pep-0257/:</p> <p>"A docstring is a string literal that occurs as the first statement in a module, function, class, or method definition. Such a docstring becomes the <code>__doc__</code> special attribute of that object.</p> <p>All modules should normally have docstrings, and all functions and classes exported by a module should also have docstrings. Public methods (including the <code>__init__</code> constructor) should also have docstrings. A package may be documented in the module docstring of the <code>__init__.py</code> file in the package directory.</p> <p>String literals occurring elsewhere in Python code may also act as documentation. They are not recognized by the Python bytecode compiler and are not accessible as runtime object attributes (i.e. not assigned to <code>__doc__</code>), but two types of extra docstrings may be extracted by software tools:</p> <ul style="list-style-type: none">▪ String literals occurring immediately after a simple assignment at the top level of a module, class, or <code>__init__</code> method are called "attribute docstrings".▪ String literals occurring immediately after another docstring are called "additional docstrings". <p>Please see PEP 258, "Docutils Design Specification" [2], for a detailed description of attribute and additional docstrings."</p>
Dropbox	<p>From www.dropbox.com:</p> <p>"Dropbox is a free service that lets you bring all your photos, docs, and videos anywhere. Any file you save to your Dropbox will also automatically save to all your computers, phones, and even the Dropbox website. This means that you can start working on your computer at school or the office, and finish on your home computer. Never email yourself a file again!"</p>

TERM	DEFINITION
Embedded Systems	<p>Excerpt From Wikipedia, the free encyclopedia</p> <p>"An embedded system is a computer system with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts. Embedded systems control many devices in common use today. Ninety-eight percent of all microprocessors are manufactured as components of embedded systems.</p> <p>Examples of properties of typically embedded computers when compared with general-purpose counterparts are low power consumption, small size, rugged operating ranges, and low per-unit cost. This comes at the price of limited processing resources, which make them significantly more difficult to program and to interact with. However, by building intelligence mechanisms on top of the hardware, taking advantage of possible existing sensors and the existence of a network of embedded units, one can both optimally manage available resources at the unit and network levels as well as provide augmented functions, well beyond those available. For example, intelligent techniques can be designed to manage power consumption of embedded systems.</p> <p>Modern embedded systems are often based on microcontrollers (i.e. CPUs with integrated memory or peripheral interfaces), but ordinary microprocessors (using external chips for memory and peripheral interface circuits) are also common, especially in more-complex systems. In either case, the processor(s) used may be types ranging from general purpose to those specialized in certain class of computations, or even custom designed for the application at hand. A common standard class of dedicated processors is the digital signal processor (DSP).</p> <p>Since the embedded system is dedicated to specific tasks, design engineers can optimize it to reduce the size and cost of the product and increase the reliability and performance. Some embedded systems are mass-produced, benefiting from economies of scale.</p> <p>Embedded systems range from portable devices such as digital watches and MP3 players, to large stationary installations like traffic lights, factory controllers, and largely complex systems like hybrid vehicles, MRI, and avionics. Complexity varies from low, with a single microcontroller chip, to very high with multiple units, peripherals and networks mounted inside a large chassis or enclosure."</p>
Extension Module	<p>A module written in the low-level language of the Python implementation: C/C++ for Python, Java for Jython. Typically contained in a single dynamically loadable pre-compiled file, e.g. a shared object (.so) file for Python extensions on Unix, a DLL (given the .pyd extension) for Python extensions on Windows, or a Java class file for Jython extensions. (Note that currently, the Distutils only handles C/C++ extensions for Python.)</p>
FreeBSD	<p>From Wikipedia, the free encyclopedia:</p> <p>"FreeBSD is a free Unix-like operating system descended from Research Unix via Berkeley Software Distribution (BSD). Although for legal reasons FreeBSD cannot use the Unix trademark, it is a direct descendant of BSD, which was historically also called "BSD Unix" or "Berkeley Unix". The first version of FreeBSD was released in 1993, and today FreeBSD is the most widely used open-source BSD distribution, accounting for more than three-quarters of all installed systems running open-source BSD derivatives.[3]</p> <p>FreeBSD has similarities with Linux, with two major differences in scope and licensing: FreeBSD maintains a complete operating system, i.e. the project delivers kernel, device drivers, userland utilities and documentation, as opposed to a kernel only;[4] and FreeBSD source code is generally released under a permissive BSD license as opposed to the more restrictive GPL.</p> <p>The FreeBSD project includes a security team overlooking all software shipped in the base distribution. A wide range of additional third-party applications may be installed via two package managers, "pkgng" and the FreeBSD Ports, or by directly compiling source code. Due to its</p>

TERM	DEFINITION
	<p>permissive licensing terms, much of FreeBSD's code base has become an integral part of other operating systems such as Juniper JUNOS and Apple's OS X."</p>
Functional Requirements	<p>From Wikipedia, the free encyclopedia:</p> <p>"In software engineering (and System Engineering), a functional requirement defines a function of a system or its component. A function is described as a set of inputs, the behavior, and outputs (see also software). Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all the cases where the system uses the functional requirements are captured in use cases. Functional requirements are supported by non-functional requirements (also known as quality requirements), which impose constraints on the design or implementation (such as performance requirements, security, or reliability). Generally, functional requirements are expressed in the form "system must do <requirement>", while non-functional requirements are "system shall be <requirement>". The plan for implementing functional requirements is detailed in the system design. The plan for implementing non-functional requirements is detailed in the system architecture.</p> <p>As defined in requirements engineering, functional requirements specify particular results of a system. This should be contrasted with non-functional requirements which specify overall characteristics such as cost and reliability. Functional requirements drive the application architecture of a system, while non-functional requirements drive the technical architecture of a system.</p> <p>In some cases a requirements analyst generates use cases after gathering and validating a set of functional requirements. The hierarchy of functional requirements is: user/stakeholder request → feature → use case → business rule. Each use case illustrates behavioral scenarios through one or more functional requirements. Often, though, an analyst will begin by eliciting a set of use cases, from which the analyst can derive the functional requirements that must be implemented to allow a user to perform each use case."</p>
GNU	<p>From Wikipedia, the free encyclopedia:</p> <p>"GNU [2][3] is a Unix-like computer operating system developed by the GNU Project, ultimately aiming to be a "complete Unix-compatible software system" [1][4][5] composed wholly of free software. Development of GNU was initiated by Richard Stallman in 1983 [1][6] and was the original focus of the Free Software Foundation (FSF). [1][7][8][9] Non-GNU kernels, most famously the Linux kernel, can also be used with GNU. [10][11][12] The FSF maintains that Linux, when used with GNU tools and utilities, should be considered a variant of GNU, and promotes the term Linux for such systems (leading to the Linux naming controversy). [13][14][15]</p> <p>GNU is a recursive acronym for "GNU's Not Unix!", [1][16] chosen because GNU's design is Unix-like, but differs from Unix by being free software and containing no Unix code. [17] Programs released under the auspices of the GNU Project are called GNU packages or GNU programs. The system's basic components include the GNU Compiler Collection (GCC), the GNU C library (glibc), and GNU Core Utilities (coreutils), [1] but also the GNU Debugger (GDB), GNU Binary Utilities (binutils), and the bash shell. [18] GNU developers have contributed GNU/Linux Linux ports of GNU applications and utilities, which are now also widely used on other operating systems such as BSD variants, Solaris and Mac OS X. [19]</p> <p>The GNU General Public License (GPL), the GNU Lesser General Public License (LGPL), and the GNU Free Documentation License (GFDL) were written for GNU and the GNU Affero General Public License was written as an extended version of GPL version 3 for programs run over a network, but the GNU Project's licenses are also used by many unrelated projects. A minority of the software used by GNU, such as the X Window System, is licensed under permissive free software licenses.</p>

TERM	DEFINITION
	Richard Stallman views GNU as a "technical means to a social end".[20]"
GNU/Linux	<p>Excerpted From https://www.gnu.org/gnu/linux-and-gnu.html :</p> <p>"Linux and the GNU System by Richard Stallman</p> <p>For more information see also the GNU/Linux FAQ, and Why GNU/Linux?</p> <p>Many computer users run a modified version of the GNU system every day, without realizing it. Through a peculiar turn of events, the version of GNU which is widely used today is often called "Linux", and many of its users are not aware that it is basically the GNU system, developed by the GNU Project.</p> <p>There really is a Linux, and these people are using it, but it is just a part of the system they use. Linux is the kernel: the program in the system that allocates the machine's resources to the other programs that you run. The kernel is an essential part of an operating system, but useless by itself; it can only function in the context of a complete operating system. Linux is normally used in combination with the GNU operating system: the whole system is basically GNU with Linux added, or GNU/Linux. All the so-called ?Linux? distributions are really distributions of GNU/Linux.</p> <p>Many users do not understand the difference between the kernel, which is Linux, and the whole system, which they also call ?Linux?. The ambiguous use of the name doesn't help people understand. These users often think that Linus Torvalds developed the whole operating system in 1991, with a bit of help.</p> <p>Programmers generally know that Linux is a kernel. But since they have generally heard the whole system called ?Linux? as well, they often envisage a history that would justify naming the whole system after the kernel. For example, many believe that once Linus Torvalds finished writing Linux, the kernel, its users looked around for other free software to go with it, and found that (for no particular reason) most everything necessary to make a Unix-like system was already available.</p> <p>What they found was no accident?it was the not-quite-complete GNU system. The available free software added up to a complete system because the GNU Project had been working since 1984 to make one. In the The GNU Manifesto we set forth the goal of developing a free Unix-like system, called GNU. The Initial Announcement of the GNU Project also outlines some of the original plans for the GNU system. By the time Linux was started, GNU was almost finished.</p> <p>Most free software projects have the goal of developing a particular program for a particular job. For example, Linus Torvalds set out to write a Unix-like kernel (Linux); Donald Knuth set out to write a text formatter (TeX); Bob Scheifler set out to develop a window system (the X Window System). It's natural to measure the contribution of this kind of project by specific programs that came from the project.</p> <p>If we tried to measure the GNU Project's contribution in this way, what would we conclude? One CD-ROM vendor found that in their "Linux distribution", GNU software was the largest single contingent, around 28% of the total source code, and this included some of the essential major components without which there could be no system. Linux itself was about 3%. (The proportions in 2008 are similar: in the "main" repository of gNewSense, Linux is 1.5% and GNU packages are 15%.) So if you were going to pick a name for the system based on who wrote the programs in the system, the most appropriate single choice would be "GNU".</p> <p>But that is not the deepest way to consider the question. The GNU Project was not, is not, a project to develop specific software packages. It was not a project to develop a C compiler, although we did that. It was not a project to develop a text editor, although we developed one. The GNU Project set out to develop a complete free Unix-like system: GNU."</p> <p><i>The complete article is available at the aforementioned link.</i></p>

TERM	DEFINITION
gnuwin32	<p>From Wikipedia, the free encyclopedia:</p> <p>"The GnuWin32 project provides native ports in the form of runnable computer programs, patches, and source code for various GNU and open source tools and software, much of it modified to run on the 32-bit Windows platform. The ports included in the GnuWin32 packages are:</p> <ul style="list-style-type: none"> ▪ GNU utilities such as bc, bison, chess, Coreutils, diffutils, ed, Flex, gawk, gettext, grep, Groff, gzip, iconv, less, m4, patch, readline, rx, sharutils, sed, tar, texinfo, units, Wget, which ▪ Archive management and compression tools, such as: arc, arj, bzip2, gzip, lha, zip, zlib. ▪ Non-GNU utilities such as: cygutils, file, ntfspgms, OpenSSL, PCRE. ▪ Graphics tools. ▪ PDCurses ▪ Tools for processing text. ▪ Mathematical software and statistics Software." <p>See also:</p> <ul style="list-style-type: none"> ▪ Cygwin ▪ DJGPP ▪ GNUWin II ▪ Microsoft Windows Services for UNIX ▪ MinGW, MSYS ▪ UnxUtils ▪ UWIN
GUI	<p>Acronym for Graphical User Interface.</p> <p>From Wikipedia, the free encyclopedia:</p> <p>"In computing, a graphical user interface (GUI, commonly pronounced gooey[1]) is a type of user interface that allows users to interact with electronic devices with images rather than text commands. GUIs can be used in computers, hand-held devices such as MP3 players, portable media players or gaming devices, household appliances and office equipment. A GUI represents the information and actions available to a user through graphical icons and visual indicators such as secondary notation, as opposed to text-based interfaces, typed command labels or text navigation. The actions are usually performed through direct manipulation of the graphical elements.[2]"</p>
High-level API	<p>A programming interface provides more functionality within one statement than a lower-level interface. High-level interfaces are designed to enable the programmer to write code in a shorter amount of time and to be less involved with the details of the software module or hardware device that is performing the required services.</p>
Hypervisor	<p>From Wikipedia, the free encyclopedia</p> <p>"In computing, a hypervisor, also called virtual machine manager (VMM), is one of many hardware virtualization techniques allowing multiple operating systems, termed guests, to run concurrently on a host computer. It is so named because it is conceptually one level higher than a supervisory program. The hypervisor presents to the guest operating systems a virtual operating platform and manages the execution of the guest operating systems. Multiple instances of a variety of operating systems may share the virtualized hardware resources. Hypervisors are very commonly installed on server hardware, with the function of running guest operating systems, that themselves act as servers.</p> <p>The term can be used to describe the interface provided by the specific cloud computing</p>

TERM	DEFINITION
	<p>functionality infrastructure as a service (IaaS).[1][2]</p> <p>The term "hypervisor" was first used in 1965, referring to software that accompanied an IBM RPQ for the IBM 360/65. It allowed the model IBM 360/65 to share its memory: half acting as an IBM 360 and half as an emulated IBM 7080. The software, labeled "hypervisor," did the switching between the two modes on split-time basis. The term hypervisor was coined as an evolution of the term "supervisor," the software that provided control on earlier hardware.[3][4]"</p>
Instrumentation	<p>The application of technology for the measurement and control of process variables within a production or manufacturing area. An instrument is a device that measures a physical quantity such as flow, temperature, level, distance, angle, or pressure. Instruments may be as simple as direct reading thermometers or may be complex multi-variable process analyzers. Instruments are often part of a control system in refineries, factories, and vehicles.</p>
Interface Requirements Specification	<p>The Interface Requirements Specification (IRS) specifies the requirements imposed on one or more systems, subsystems, Hardware Configuration Items (HWCIs), Computer Software Configuration Items (CSCIs), manual operations, or other system components to achieve one or more interfaces among these entities. An IRS can cover any number of interfaces.</p>
Linux	<p>From Wikipedia, the free encyclopedia</p> <p>"This article is about the operating system. For the kernel, see Linux kernel.</p> <p>Linux [8][9][10] is a Unix-like computer operating system assembled under the model of free and open source software development and distribution. The defining component of GNU/Linux Linux is the Linux kernel, an operating system kernel first released 5 October 1991 by Linus Torvalds.[11][12]"</p> <p>NOTE:</p> <ul style="list-style-type: none"> ▪ The Free Software Foundation (FSF) is responsible for the GNU Project and maintains that Linux Linux, when used with GNU tools and utilities, should be considered a variant of GNU, and promotes the term Linux for such systems (leading to the Linux naming controversy).
Low-level API	<p>A programming interface that is the most detailed, allowing the programmer to manipulate functions within a software module or within hardware at a very granular level.</p>
macOS / MAC OS X	<p>From Wikipedia, the free encyclopedia</p> <p>"Mac OS is a series of graphical user interface-based operating systems developed by Apple Inc. (formerly Apple Computer, Inc.) for their Macintosh line of computer systems. Mac OS is credited with popularizing the graphical user interface. The original form of what Apple would later name the "Mac OS" (currently OSX) was the integral and unnamed system software first introduced in 1984 with the original Macintosh, usually referred to simply as the System software."</p>
ManPage	<p>A manpage (short for manual page) is a form of online software documentation usually found on a Unix or Unix-like operating system. Topics covered include computer programs (including library and system calls), formal standards and conventions, and even abstract concepts.</p>

TERM	DEFINITION
Massachusetts Institute of Technology License	<p>From Wikipedia, the free encyclopedia</p> <p>"The MIT License is a free software license originating at the Massachusetts Institute of Technology (MIT). It is a permissive free software license, meaning that it permits reuse within proprietary software provided all copies of the licensed software include a copy of the MIT License terms. Such proprietary software retains its proprietary nature even though it incorporates software under the MIT License. The license is also GPL-compatible, meaning that the GPL permits combination and redistribution with software that uses the MIT License.[1]</p> <p>Notable software packages that use one of the versions of the MIT License include Expat, PuTTY, the Mono development platform class libraries, Ruby on Rails, Lua (from version 5.0 onwards), Wayland and the X Window System, for which the license was written."</p>
Microsoft Windows	<p>From Wikipedia, the free encyclopedia</p> <p>"Microsoft Windows is a series of graphical interface operating systems developed, marketed, and sold by Microsoft.</p> <p>Microsoft introduced an operating environment named Windows on November 20, 1985 as an add-on to MS-DOS in response to the growing interest in graphical user interfaces (GUIs).[2] Microsoft Windows came to dominate the world's personal computer market with over 90% market share, overtaking Mac OS, which had been introduced in 1984."</p>
MIL-STD-498	<p>From Wikipedia, the free encyclopedia</p> <p>"MIL-STD-498 (Military-Standard-498) was a United States military standard whose purpose was to "establish uniform requirements for software development and documentation." It was released Nov. 8, 1994, and replaced DOD-STD-2167A, DOD-STD-7935A, and DOD-STD-1703. It was meant as an interim standard, to be in effect for about two years until a commercial standard was developed.</p> <p>Unlike previous efforts like the seminal "2167A" which was mainly focused on the risky new area of software development, "498" was the first attempt at a truly comprehensive description of the systems development life-cycle. It was the baseline that all of the ISO, IEEE, and related efforts after it replaced. It also contains much of the material that the subsequent professionalization of project management covered in the Project Management Body of Knowledge (PMBOK). The document "MIL-STD-498 Overview and Tailoring Guidebook" is 98 pages. The "MIL-STD-498 Application and Reference Guidebook" is 516 pages. Associated to these were document templates, or Data Item Descriptions, described below, bringing documentation and process order that could scale to projects of the size humans were then conducting (aircraft, battleships, canals, dams, factories, satellites, submarines, etcetera).</p> <p>It was one of the few military standards that survived the "Perry Memo", then U.S. Secretary of Defense William Perry's 1994 memorandum commanding the discontinuation of defense standards. However, it was canceled on May 27, 1998 and replaced by the essentially identical demilitarized version EIA J-STD-016[1] [2] as a process example guide for IEEE 12207. Several programs outside of the U.S. military continued to use the standard due to familiarity and perceived advantages over alternative standards, such as free availability of the standards documents and presence of process detail including contractually-usable Data Item Descriptions."</p> <p>From http://everyspec.com/MIL-STD/MIL-STD-0300-0499/MIL-STD-498_25500/</p> <p>"MIL-STD-498, MILITARY STANDARD: SOFTWARE DEVELOPMENT AND DOCUMENTATION (05 DEC 1994) [SUPERSEDING MIL-STD-2167A, DOD-STD-7935A & DOD-STD-1703] [S/S BY IEEE/EIA 12207.0, IEEE/EIA 12207.1 & IEEE/EIA 12207.2]., The purpose of this standard is to establish uniform requirements for software development and documentation. This standard merges DOD-STD-2167A</p>

TERM	DEFINITION
	<p>and DOD-STD-7935A to define a set of activities and documentation suitable for the development of both weapon systems and Automated Information Systems. A conversion guide from these standards to MIL-STD-498 is provided in Appendix I. Other changes include improved compatibility with incremental and evolutionary development models; improved compatibility with non-hierarchical design methods; improved compatibility with computer-aided software engineering (CASE) tools; alternatives to, and more flexibility in, preparing documents; clearer requirements for incorporating reusable software; introduction of software management indicators; added emphasis on software supportability; and improved links to systems engineering. This standard supersedes DOD-STD-2167A, DOD-STD- 7935A, and DOD-STD-1703 (NS)."</p> <p>A copy of the specification is available via a download link.</p>
Module	The basic unit of code reusability in Python: a block of code imported by some other code. Three types of modules concern us here: pure Python modules, extension modules, and packages.
nCurses	<p>From Wikipedia, the free encyclopedia</p> <p>"ncurses (new curses) is a programming library that provides an API which allows the programmer to write text-based user interfaces in a terminal-independent manner. It is a toolkit for developing "GUI-like" application software that runs under a terminal emulator. It also optimizes screen changes, in order to reduce the latency experienced when using remote shells."</p>
Non-Functional Requirements	<p>From Wikipedia, the free encyclopedia:</p> <p>See Functional Requirements for definition and relationship.</p>
Notebooks	A collection of commentaries that express opinions or offerings of explanations about events or situations that might be useful to Toolkit installers, developers, operators, troubleshooters and distributors. The documents may be in Application-specific formats (Adobe PDF, JPEG Bit-mapped image, Microsoft Office, Plain text etc.).
OpenIndiana	<p>From Wikipedia, the free encyclopedia</p> <p>"OpenIndiana is a Unix-like computer operating system released as free and open source software. It forked from OpenSolaris after the discontinuation of that project by Oracle[1] and aims to continue development and distribution of the OpenSolaris codebase.[2] The project operates under the umbrella of the Illumos Foundation.[2] The stated aim of the project is "[...] to become the defacto OpenSolaris distribution installed on production servers where security and bug fixes are required free of charge".[3]"</p>
OpenSolaris	<p>From Wikipedia, the free encyclopedia</p> <p>"OpenSolaris (...) was an open source computer operating system based on Solaris created by Sun Microsystems. It was also the name of the project initiated by Sun to build a developer and user community around the software. After the acquisition of Sun Microsystems in 2010, Oracle decided to discontinue open development of the core software, and replaced the OpenSolaris distribution model with the proprietary Solaris Express.</p> <p>Prior to Oracle's moving of core development "behind closed doors", a group of former OpenSolaris developers decided to "fork" the core software under the name OpenIndiana. The project, a part of the Illumos Foundation, aims to continue the development and distribution of the OpenSolaris codebase.[5]</p> <p>OpenSolaris is a descendant of the UNIX System V Release 4 (SVR4) code base developed by Sun and AT&T in the late 1980s. It is the only version of the System V variant of UNIX available as</p>

TERM	DEFINITION
	<p>open source.[6] OpenSolaris is developed as a combination of several software consolidations that were open sourced subsequent to Solaris 10. It includes a variety of free software, including popular desktop and server software.[7][8] On Friday, August 13, 2010, details started to emerge relating to the restructuring of the OpenSolaris project, the pending release of the new future commercial version of Solaris, Solaris 11, and how open source community interactions are being adjusted.[9]"</p>
Package	<p>A module that contains other modules; typically contained in a directory in the file system and distinguished from other directories by the presence of a file <code>__init__.py</code>.</p>
Parallels Desktop for Mac	<p>From Wikipedia, the free encyclopedia</p> <p>"Parallels Desktop for Mac by Parallels, Inc., is software providing hardware virtualization for Macintosh computers with Intel processors.</p> <p>Technical</p> <p>Parallels Desktop for Mac is a hardware emulation virtualization software, using hypervisor technology that works by mapping the host computer's hardware resources directly to the virtual machine's resources. Each virtual machine thus operates identically to a standalone computer, with virtually all the resources of a physical computer.[3] Because all guest virtual machines use the same hardware drivers irrespective of the actual hardware on the host computer, virtual machine instances are highly portable between computers. For example, a running virtual machine can be stopped, copied to another physical computer, and restarted."</p> <p>Parallels Tools</p> <p>Parallels Tools are a suite of behind-the-scenes tools that allow seamless operating between Mac OS X and Windows or another guest operating system.</p> <p>Each Parallels release includes its own Parallels Tools. Those tools interface the Guest Operating System's Virtual Machine to the Parallels Desktop installed on the host computer. The Tools enable the Host and Guest OS to share resources.</p> <p>After upgrading the Parallels Desktop from 9 to 10 on one host computer it was still possible to run recent copies of the Parallels 10 Guest OS virtual machines on a host computer that could not be upgraded to Parallels 10 simply by:</p> <ul style="list-style-type: none"> ▪ Attaching a hard drive with the Parallels 10 Guest OS virtual machine. ▪ Manually changing the Parallels Guest OS configuration to suit the available host memory and devices. ▪ Allowing Parallels to automatically (or manually initiating) re-install of the available older Parallels (8 or 9) Tools. ▪ Launching the re-configured Parallels Guest OS.
PC-BSD or PCBSD	<p>From Wikipedia, the free encyclopedia</p> <p>"PC-BSD, or PCBSD, is a Unix-like, desktop-oriented operating system built upon the most recent releases of FreeBSD. It aims to be easy to install by using a graphical installation program, and easy and ready-to-use immediately by providing KDE SC, LXDE, Xfce, and MATE [1] as the graphical user interface. It provides official binary nVidia and Intel drivers for hardware acceleration and an optional 3D desktop interface through Kwin, and Wine is ready-to-use in running Microsoft Windows software. PC-BSD is able to run Linux software,[2] in addition to FreeBSD ports, and it has its own package management system that allows users to graphically install pre-built software packages from a single downloaded executable file, which is unique for BSD operating systems.</p> <p>PC-BSD supports ZFS, and the installer offers disk encryption with geli so the system will require a passphrase before booting."</p>

TERM	DEFINITION
PDCurses	<p>PDCurses is an implementation of the curses library for X11. It provides the ability for existing text-mode curses programs to be re-built as native X11 applications with very little modification. PDCurses for X11 is also known as XCurseS.</p> <p>It is available from http://pdcurses.sourceforge.net. Version 2.6 enables Python applications launched within the Windows Command Prompt shell to import and use the Python Curses module.</p> <p>However, Version 2.6 cannot be used by the "tsWxGraphicalTextUserInterface" module to emulate "wxPython" because it lacks the mouse button definitions and interface features available with Unix-like shells such as bash.</p>
POSIX	<p>From Wikipedia, the free encyclopedia</p> <p>"Not to be confused with Unix, Unix-like, or Linux.</p> <p>An acronym for "Portable Operating System Interface", is a family of standards specified by the IEEE for maintaining compatibility between operating systems. POSIX defines the application programming interface (API), along with command line shells and utility interfaces, for software compatibility with variants of Unix and other operating systems.[1][2]"</p>
Programming Tools or Software Development Tools	<p>From Wikipedia, the free encyclopedia</p> <p>"A programming tool or software development tool is a computer program that software developers use to create, debug, maintain, or otherwise support other programs and applications. The term usually refers to relatively simple programs, that can be combined together to accomplish a task, much as one might use multiple hand tools to fix a physical object. The ability to use a variety of tools productively is one hallmark of a skilled software engineer.</p> <p>The most basic tools are a source code editor and a compiler or interpreter, which are used ubiquitously and continuously. Other tools are used more or less depending on the language, development methodology, and individual engineer, and are often used for a discrete task, like a debugger or profiler. Tools may be discrete programs, executed separately – often from the command line – or may be parts of a single large program, called an integrated development environment (IDE). In many cases, particularly for simpler use, simple ad hoc techniques are used instead of a tool, such as print debugging instead of using a debugger, manual timing (of overall program or section of code) instead of a profiler, or tracking bugs in a text file or spreadsheet instead of a bug tracking system.</p> <p>The distinction between tools and applications is murky. For example, developers use simple databases (such as a file containing a list of important values) all the time as tools.[dubious – discuss] However a full-blown database is usually thought of as an application or software in its own right. For many years, computer-assisted software engineering (CASE) tools were sought after. Successful tools have proven elusive.[citation needed] In one sense, CASE tools emphasized design and architecture support, such as for UML. But the most successful of these tools are IDEs."</p>
Pure Python Module	<p>A module written in Python and contained in a single .py file (and possibly associated .pyc and/or .pyo files). Sometimes referred to as a "pure module."</p>
Python	<p>From Wikipedia, the free encyclopedia:</p> <p>"Python is a general-purpose, high-level programming language[11] whose design philosophy emphasizes code readability.[12] Python claims to combine "remarkable power with very clear syntax",[13] and its standard library is large and comprehensive.</p> <p>Python supports multiple programming paradigms, primarily but not limited to object-oriented, imperative and, to a lesser extent, functional programming styles. It features a fully dynamic type system and automatic memory management, similar to that of Scheme, Ruby, Perl, and Tcl. Like other dynamic languages, Python is often used as a scripting language, but is also used in a wide</p>

TERM	DEFINITION
	<p>range of non-scripting contexts. Using third-party tools, Python code can be packaged into standalone executable programs. Python interpreters are available for many operating systems.</p> <p>The reference implementation of Python (CPython) is free and open source software and has a community-based development model, as do all or nearly all of its alternative implementations. CPython is managed by the non-profit Python Software Foundation."</p>
Python Software Foundation License	<p>From Wikipedia, the free encyclopedia:</p> <ul style="list-style-type: none"> ▪ "Python Software Foundation License FSF approved Yes [1] ▪ OSI approved Yes ▪ GPL compatible Yes [1] ▪ Copyleft No <p>The Python Software Foundation License (PSFL) is a BSD-style, permissive free software license which is compatible with the GNU General Public License (GPL).[1] Its primary use is for distribution of the Python project software. Unlike the GPL the Python license is not a copyleft license, and allows modifications to the source code, as well as the construction of derivative works, without making the code open-source. The PSFL is listed as approved on both FSF's approved licenses list,[1] and OSI's approved licenses list.</p> <p>Earlier versions of Python were under the so-called Python License, which is incompatible with the GPL. The reason given for this incompatibility by Free Software Foundation was that "this Python license is governed by the laws of the 'State of Virginia', in the USA", and the GPL does not permit this.[2]</p> <p>The year that Python's creator Guido van Rossum changed the license to fix this incompatibility, he was awarded the Free Software Foundation Award for the Advancement of Free Software.[3]"</p>
Python Virtual Machine	<p>A virtual machine designed to interpret and execute programs implemented in the Python programming language.</p> <p>From Wikipedia, the free encyclopedia:</p> <p>"A virtual machine (VM) is a software implementation of a machine (i.e. a computer) that executes programs like a physical machine. Virtual machines are separated into two major categories, based on their use and degree of correspondence to any real machine. A system virtual machine provides a complete system platform which supports the execution of a complete operating system (OS). In contrast, a process virtual machine is designed to run a single program, which means that it supports a single process. An essential characteristic of a virtual machine is that the software running inside is limited to the resources and abstractions provided by the virtual machine—it cannot break out of its virtual world.</p> <p>A virtual machine was originally defined by Popek and Goldberg as "an efficient, isolated duplicate of a real machine". Current use includes virtual machines which have no direct correspondence to any real hardware.[2]"</p>
Repository	<p>Excerpt From Wikipedia, the free encyclopedia:</p> <p>"A software repository is a storage location from which software packages may be retrieved and installed on a computer."</p> <p>The TeamSTARS "tsWxGTUI_PyVx" Toolkit repository is the collection of documentation and computer program source code files that is being distributed by its author in order to publish and share the intellectual property with others.</p>

TERM	DEFINITION
Root Package	The root of the hierarchy of packages. (This isn't really a package, since it doesn't have an <code>__init__.py</code> file. But we have to call it something.) The vast majority of the standard library is in the root package, as are many small, standalone third-party modules that don't belong to a larger module collection. Unlike regular packages, modules in the root package can be found in many directories: in fact, every directory listed in <code>sys.path</code> contributes modules to the root package.
Simulation	The application of technology for the imitation of the operation of a real-world process or system over time. The act of simulating something first requires that a model be developed; this model represents the key characteristics or behaviors/functions of the selected physical or abstract system or process. The model represents the system itself, whereas the simulation represents the operation of the system over time.
Site-Package	The location where third-party packages are installed (i.e., those not part of the core Python distribution). NOTE: That with Linux, Mac OS X and Unix operating systems one must have root privileges to write to that location. Unlike the contents of the Developer-Sandbox, the third-party site-package and its users must explicitly import via the <code>site-package.package.module</code> path identifier.
Software Engineer	An individual who will specify, plan, supervise and/or perform the design, development, coding, testing, debugging, maintenance and support of application programs, with Command Line Interface or Graphical User Interface, to be used by System Operators. The term also applies to those individuals who perform similar engineering activities associated with communication, data base, simulation, operating system, device driver, test and diagnostic components.
Software Requirements Specification	The Software Requirements Specification (SRS) specifies the requirements for a Computer Software Configuration Item (CSCI) and the methods to be used to ensure that each requirement has been met. Requirements pertaining to the CSCI's external interfaces may be presented in the SRS or in one or more Interface Requirements Specifications (IRs) (DI-IPSC-81434) referenced from the SRS.

TERM	DEFINITION
Solaris	<p>From Wikipedia, the free encyclopedia</p> <p>"Solaris is a Unix operating system originally developed by Sun Microsystems. It superseded their earlier SunOS in 1993. Oracle Solaris, as it is now known, has been owned by Oracle Corporation since Oracle's acquisition of Sun in January 2010.[2]</p> <p>Solaris is known for its scalability, especially on SPARC systems, and for originating many innovative features such as DTrace, ZFS and Time Slider.[3][4] Solaris supports SPARC-based and x86-based workstations and servers from Sun and other vendors, with efforts underway to port to additional platforms. Solaris is registered as compliant with the Single Unix Specification.</p> <p>Solaris was historically developed as proprietary software, then in June 2005 Sun Microsystems released most of the codebase under the CDDL license, and founded the OpenSolaris open source project.[5] With OpenSolaris, Sun wanted to build a developer and user community around the software. After the acquisition of Sun Microsystems in January 2010, Oracle decided to discontinue the OpenSolaris distribution and the development model.[6][7] Just ten days before the internal Oracle memo announcing this decision to employees was "leaked", Garrett D'Amore had announced[8] the illumos project, creating a fork of the Solaris kernel and launching what has since become a thriving alternative to Oracle Solaris.</p> <p>In August 2010, Oracle discontinued providing public updates to the source code of the Solaris Kernel, effectively turning Solaris 11 into a closed source proprietary operating system. However, through the Oracle Technology Network (OTN), industry partners can still gain access to the in-development Solaris source code.[7] The Open source portion of Solaris 11 is available for download from Oracle.[9]"</p>
Source Code	<p>Excerpt From Wikipedia, the free encyclopedia:</p> <p>"In computing, source code is any collection of computer instructions (possibly with comments) written using some human-readable computer language, usually as text. The source code of a program is specially designed to facilitate the work of computer programmers, who specify the actions to be performed by a computer mostly by writing source code. The source code is often transformed by a compiler program into low-level machine code understood by the computer. The machine code might then be stored for execution at a later time. Alternatively, an interpreter can be used to analyze and perform the outcomes of the source code program directly on the fly.</p> <p>Most computer applications are distributed in a form that includes executable files, but not their source code. If the source code were included, it would be useful to a user, programmer, or system administrator, who may wish to modify the program or to understand how it works.</p> <p>Aside from its machine-readable forms, source code also appears in books and other media; often in the form of small code snippets, but occasionally complete code bases; a well-known case is the source code of PGP."</p>
Stakeholder	Those persons, groups or organizations with an interest in a project.
Supervisory Control And Data Acquisition	<p>Excerpt From Wikipedia, the free encyclopedia</p> <p>"Supervisory control and data acquisition (SCADA) is a system for remote monitoring and control that operates with coded signals over communication channels (using typically one communication channel per remote station).</p> <p>The control system may be combined with a data acquisition system by adding the use of coded signals over communication channels to acquire information about the status of the remote equipment for display or for recording functions. It is a type of industrial control system (ICS). Industrial control systems are computer-based systems that monitor and control industrial processes that exist in the physical world. SCADA systems historically distinguish themselves from other ICS systems by being large-scale processes that can include multiple sites, and large distances. These</p>

TERM	DEFINITION
	<p>processes include industrial, infrastructure, and facility-based processes, as described below:</p> <ul style="list-style-type: none"> ▪ Industrial processes include those of manufacturing, production, power generation, fabrication, and refining, and may run in continuous, batch, repetitive, or discrete modes. ▪ Infrastructure processes may be public or private, and include water treatment and distribution, wastewater collection and treatment, oil and gas pipelines, electrical power transmission and distribution, wind farms, civil defense siren systems, and large communication systems. ▪ Facility processes occur both in public facilities and private ones, including buildings, airports, ships, and space stations. They monitor and control heating, ventilation, and air conditioning systems (HVAC), access, and energy consumption. <p>Common system components</p> <p>A SCADA system usually consists of the following subsystems:</p> <ul style="list-style-type: none"> ▪ Remote terminal units (RTUs) connect to sensors in the process and convert sensor signals to digital data. They have telemetry hardware capable of sending digital data to the supervisory system, as well as receiving digital commands from the supervisory system. RTUs often have embedded control capabilities such as ladder logic in order to accomplish boolean logic operations. ▪ Programmable logic controller (PLCs) connect to sensors in the process and convert sensor signals to digital data. PLCs have more sophisticated embedded control capabilities (typically one or more IEC 61131-3 programming languages) than RTUs. PLCs do not have telemetry hardware, although this functionality is typically installed alongside them. PLCs are sometimes used in place of RTUs as field devices because they are more economical, versatile, flexible, and configurable. ▪ A telemetry system is typically used to connect PLCs and RTUs with control centers, data warehouses, and the enterprise. Examples of wired telemetry media used in SCADA systems include leased telephone lines and WAN circuits. Examples of wireless telemetry media used in SCADA systems include satellite (VSAT), licensed and unlicensed radio, cellular and microwave. ▪ A data acquisition server is a software service which uses industrial protocols to connect software services, via telemetry, with field devices such as RTUs and PLCs. It allows clients to access data from these field devices using standard protocols. ▪ A human-machine interface or HMI is the apparatus or device which presents processed data to a human operator, and through this, the human operator monitors and interacts with the process. The HMI is a client that requests data from a data acquisition server or in most installations the HMI is the graphical user interface for the operator, collects all data from external devices, creates reports, performs alarming, sends notifications, etc. ▪ A historian is a software service which accumulates time-stamped data, boolean events, and boolean alarms in a database which can be queried or used to populate graphic trends in the HMI. The historian is a client that requests data from a data acquisition server. ▪ A supervisory (computer) system, gathering (acquiring) data on the process and sending commands (control) to the SCADA system. ▪ Communication infrastructure connecting the supervisory system to the remote terminal units. ▪ Various processes and analytical instrumentation."
SWIG	<p>Excerpt From http://swig.org (http://swig.org)</p> <p>"SWIG is a software development tool that connects programs written in C and C++ with a variety of high-level programming languages. SWIG is used with different types of target languages including common scripting languages such as Javascript, Perl, PHP, Python, Tcl and Ruby. The list</p>

TERM	DEFINITION
	of supported languages also includes non-scripting languages such as C#, Common Lisp (CLISP, Allegro CL, CFFI, UFFI), D, Go language, Java including Android, Lua, Modula-3, OCAML, Octave, Scilab and R. Also several interpreted and compiled Scheme implementations (Guile, MzScheme/Racket, Chicken) are supported. SWIG is most commonly used to create high-level interpreted or compiled programming environments, user interfaces, and as a tool for testing and prototyping C/C++ software. SWIG is typically used to parse C/C++ interfaces and generate the 'glue code' required for the above target languages to call into the C/C++ code. SWIG can also export its parse tree in the form of XML and Lisp s-expressions. SWIG is free software and the code that SWIG generates is compatible with both commercial and non-commercial projects."
System Administration Utilities	Computer programs that computer system administrators use to install, debug, maintain, or otherwise support computer hardware and software.
System Administrator	An individual who will specify, plan, supervise and/or perform the installation, configuration, maintenance, repair and support of the computer hardware, operating system, application software and communication network to be used by Software Engineers and System Operators.
System Operator	An individual who will use various application programs, with associated Command Line Interface or Graphical User Interface, to interactively supervise communication, control, data base, diagnostic, instrumentation or simulation activities.
TDD	<p>Acronym for Test-Driven Development</p> <p>from: http://encyclopedia.thefreedictionary.com/Test+Driven+Development</p> <p>"Test-driven development (TDD) is a software development process that relies on the repetition of a very short development cycle: first the developer writes a failing automated test case that defines a desired improvement or new function, then produces code to pass that test and finally refactors the new code to acceptable standards. Kent Beck, who is credited with having developed or 'rediscovered' the technique, stated in 2003 that TDD encourages simple designs and inspires confidence.[1]</p> <p>Test-driven development is related to the test-first programming concepts of extreme programming, begun in 1999,[2] but more recently has created more general interest in its own right.[3]</p> <p>Programmers also apply the concept to improving and debugging legacy code developed with older techniques.[4]"</p>
tsToolKitCLI	<p>The identifier for a package of toolkit components for a Python-based Command Line Interface. The library is further subdivided into the following components:</p> <ul style="list-style-type: none"> ▪ tsLibCLI - library of command line building blocks that establishes the Unix-style, run time environment for pre-processing source files, launching application programs, handling events (registering events with date, time and event severity annotations) and configuring console terminal and file system input and output. ▪ tsTestsCLI - a set of command line interface application programs and scripts for regression testing and tutorial demos. ▪ tsToolsCLI - a set of command line interface application programs and utility scripts for: checking source code syntax and style via "plint"; generating Unix-style "man page" documentation from source code comments via "pydoc"; and installing, modifying for publication, and tracking software development metrics. ▪ tsUtilities - a library of computer system configuration, diagnostic, installation, maintenance and performance tool components for various host hardware and software platforms.
Terminal	Excerpt from Wikipedia, the free encyclopedia

TERM	DEFINITION
Emulator	"A program that emulates a video terminal within some other display architecture. Though typically synonymous with a shell or text terminal, the term terminal covers all remote terminals, including graphical interfaces. A terminal emulator inside a graphical user interface is often called a terminal window."
tsToolkitGUI	<p>The identifier for a package of toolkit components for a "wxPython"-style, "nCurses"-based Graphical-style User Interface. The library is further subdivided into the following components:</p> <ul style="list-style-type: none"> ▪ tsLibGUI - a library of graphical-style user interface building blocks that establishes the emulated run time environment for the high-level, pixel-mode, "wxPython" GUI Toolkit via the low-level, character-mode, "nCurses" Text User Interface Toolkit ▪ tsTestsGUI - a set of graphical-style user interface application programs for regression testing and tutorial demos. ▪ tsToolsGUI - a set of graphical-style user interface application programs for tracking software development metrics.
tsLibCLI	The identifier for a library of building-block components for a Python-based Command Line Interface.
tsLibGUI	The identifier for a library of building-block components for a "wxPython"-style, "nCurses"-based Graphical-style User Interface.
tsTestsCLI	The identifier for a library of qualification test components for a Python-based Command Line Interface.
tsTestsGUI	The identifier for a library of qualification test components for a "wxPython"-style, "nCurses"-based Graphical-style User Interface.
tsToolsCLI	The identifier for a library of software development, diagnostic, maintenance and project management components for a Python-based Command Line Interface.
tsToolsGUI	The identifier for a library of software development, diagnostic, maintenance and project management components for a "wxPython"-style, "nCurses"-based Graphical-style User Interface.
tsUtilities	The identifier for a library of computer system configuration, diagnostic, installation, maintenance and performance tool components for various host hardware and software platforms.
tsWxGTUI	<p>The identifier for a library of building-block components for a "wxPython"-style, "nCurses"-based Graphical-style User Interface (tsToolkitGUI) and Python-based Command Line Interface (tsToolkitCLI).</p> <p>The tsToolkitGUI and tsToolkitCLI component organization keeps the Command Line Interface components independent of the Graphical-style User Interface ones. However, it does not preclude the Graphical-style User Interface components from using the services of the Command Line Interface components as appropriate.</p>
tsWxGTUI_PyVx	<p>The generic identifier for the following Python language generation specific <i>TeamSTARS</i> "tsWxGTUI" Toolkit releases:</p> <ul style="list-style-type: none"> ▪ tsWxGTUI_Py2x --- Python 2.0.0 - 2.7.9 (Toolkit for 2nd generation Python language) ▪ tsWxGTUI_Py3x --- Python 3.0.0 - 3.4.2 (Toolkit for 3rd generation Python language)
tsWxGTUI_PyVx -Major .Minor .Maintenance	<p>The Major-Minor-Maintenance identifier for a <i>TeamSTARS</i> "tsWxGTUI" Toolkit release where:</p> <ul style="list-style-type: none"> ▪ Major --- A version number that denotes the introduction of a new design which cannot be expected to be backward compatible to a previous version. Zero (0) denotes the initial release for a product preview during its pre-alpha or beta stage. ▪ Minor --- A version number that denotes a product update that selectively introduces new

TERM	DEFINITION
	<p>features but otherwise can be expected to be backward compatible to a previous version. Zero (0) denotes the initial release.</p> <ul style="list-style-type: none"> ▪ Maintenance --- A version number that denotes a product update that corrects defects found after the release of a previous version and otherwise can be expected to be backward compatible to that previous version. Zero (0) denotes a product release in its pre-alpha stage.
UNIX	<p>From Wikipedia, the free encyclopedia</p> <p>"Unix (officially trademarked as UNIX, sometimes also written as Unix) is a multitasking, multi-user computer operating system originally developed in 1969 by a group of AT&T employees at Bell Labs, including Ken Thompson, Dennis Ritchie, Brian Kernighan, Douglas McIlroy, Michael Lesk and Joe Ossanna. The Unix operating system was first developed in assembly language, but by 1973 had been almost entirely recoded in C, greatly facilitating its further development and porting to other hardware. Today's Unix system evolution is split into various branches, developed over time by AT&T as well as various commercial vendors, universities (such as University of California, Berkeley's BSD), and non-profit organizations."</p>
Use Case	<p>Excerpt From Wikipedia, the free encyclopedia</p> <p>"In software and systems engineering, a use case is a list of action or event steps, typically defining the interactions between a role (known in the Unified Modeling Language as an actor) and a system, to achieve a goal. The actor can be a human, an external system, or time. In systems engineering, use cases are used at a higher level than within software engineering, often representing missions or stakeholder goals. The detailed requirements may then be captured in the Systems Modeling Language (SysML) or as contractual statements.</p> <p>Use case analysis is an important and valuable requirement analysis technique that has been widely used in modern software engineering since their formal introduction by Ivar Jacobson in 1992. Use case driven development is a key characteristic of many process models and frameworks such as ICONIX, the Unified Process (UP), the IBM Rational Unified Process (RUP), and the Oracle Unified Method (OUM). With its inherent iterative, incremental and evolutionary nature, use case also fits well for agile development."</p>

TERM	DEFINITION
UWIN	<p>From Wikipedia, the free encyclopedia</p> <p>"UWIN is a computer software package created by David Korn which allows programs written for the operating system Unix be built and run on Microsoft Windows with few, if any, changes. Some of the software development was subcontracted to Wipro, India. References, correct or not, to the software as U/Win and AT&T Unix for Windows can be found in some cases, especially from the early days of its existence.</p> <p>UWIN source and binaries are available under the Open Source Eclipse Public License 1.0 at AT&T AST/UWIN open source downloads."</p> <p>See also:</p> <ul style="list-style-type: none"> ▪ Cygwin, a similar project started at about the same time[2] ▪ Interix, the Microsoft product in this area ▪ Windows Services for Unix ▪ MKS Toolkit, a third-party proprietary product in this area ▪ DJGPP (DJ's GNU Programming Platform), a Windows port of Gnu programming tools ▪ Xming ▪ UnixUtils ▪ GnuWin32 ▪ GNUWin II ▪ MinGW ▪ LBW: Linux Binaries on Windows requires Interix to be installed first."
VMware Fusion	<p>From Wikipedia, the free encyclopedia</p> <p>"VMware Fusion is a software hypervisor developed by VMware for Macintosh computers with Intel processors. Fusion allows Intel-based Macs to run operating systems, such as Microsoft Windows, Linux, NetWare or Solaris on virtual machines, along with their Mac OS X operating system using a combination of paravirtualization, hardware virtualization and dynamic recompilation."</p>
VMware Workstation	<p>Excerpt From Wikipedia, the free encyclopedia:</p> <p>"VMware Workstation is a hosted hypervisor that runs on x64 versions of Windows and Linux operating systems (an x86 version of earlier releases was available); it enables users to set up virtual machines (VMs) on a single physical machine, and use them simultaneously along with the actual machine. Each virtual machine can execute its own operating system, including versions of Microsoft Windows, Linux, BSD, and MS-DOS. VMware Workstation is developed and sold by VMware, Inc., a division of Dell Technologies. There is a free-of-charge version, VMware Workstation Player, for non-commercial use. An operating systems license is needed to use proprietary ones such as Windows. Ready-made Linux VMs set up for different purposes are available from several sources.</p> <p>VMware Workstation supports bridging existing host network adapters and sharing physical disk drives and USB devices with a virtual machine. It can simulate disk drives; an ISO image file can be mounted as a virtual optical disc drive, and virtual hard disk drives are implemented as .vmdk files.</p> <p>VMware Workstation Pro can save the state of a virtual machine (a "snapshot") at any instant. These snapshots can later be restored, effectively returning the virtual machine to the saved state, as it was and free from any post-snapshot damage to the VM.</p> <p>VMware Workstation includes the ability to group multiple virtual machines in an inventory folder.</p>

TERM	DEFINITION
	The machines in such a folder can then be powered on and powered off as a single object, useful for testing complex client-server environments."
VNC	<p>From Wikipedia, the free encyclopedia:</p> <p>"In computing, Virtual Network Computing (VNC) is a graphical desktop sharing system that uses the Remote Frame Buffer protocol (RFB) to remotely control another computer. It transmits the keyboard and mouse events from one computer to another, relaying the graphical screen updates back in the other direction, over a network.[1]</p> <p>VNC is platform-independent – a VNC viewer on one operating system may connect to a VNC server on the same or any other operating system. There are clients and servers for many GUI-based operating systems and for Java. Multiple clients may connect to a VNC server at the same time. Popular uses for this technology include remote technical support and accessing files on one's work computer from one's home computer, or vice versa.</p> <p>VNC was originally developed at the Olivetti & Oracle Research Lab in Cambridge, United Kingdom. The original VNC source code and many modern derivatives are open source under the GNU General Public License.</p> <p>VNC in KDE 3.1</p> <p>There are a number of variants of VNC[2] which offer their own particular functionality; e.g., some optimized for Microsoft Windows, or offering file transfer (not part of VNC proper), etc. Many are compatible (without their added features) with VNC proper in the sense that a viewer of one flavor can connect with a server of another; others are based on VNC code but not compatible with standard VNC.</p> <p>VNC and RFB are registered trademarks of RealVNC Ltd. in the U.S. and in other countries."</p>
vt100	<p>From Wikipedia, the free encyclopedia:</p> <p>"The VT100 is a video terminal that was made by Digital Equipment Corporation (DEC). Its detailed attributes became the de facto standard for terminal emulators to emulate.</p> <p>History</p> <p>It was introduced in August 1978, following its predecessor, the VT52, and communicated with its host system over serial lines using the ASCII character set and control sequences (a.k.a. escape sequences) standardized by ANSI. The VT100 was also the first Digital mass-market terminal to incorporate "graphic renditions" (blinking, bolding, reverse video, and underlining) as well as a selectable 80 or 132 column display. All setup of the VT100 was accomplished using interactive displays presented on the screen; the setup data was stored in non-volatile memory within the terminal. The VT100 also introduced an additional character set that allowed the drawing of on-screen forms.</p> <p>The control sequences used by the VT100 family are based on the ANSI X3.64 standard, also known as ECMA-48 and ISO/IEC 6429. These are sometimes referred to as ANSI escape codes. The VT100 was not the first terminal to be based on X3.64—The Heath Company had a microprocessor-based video terminal, the Heathkit H-19 (H19), that implemented a subset of the standard proposed by ANSI in X3.64.[1] In addition, the VT100 provided backwards compatibility for VT52 users, with support for the VT52 control sequences.[2]</p> <p>In 1983, the VT100 was replaced by the more-powerful VT200 series terminals such as the VT220. In August 1995 the terminal business of Digital was sold to Boundless Technologies.[3]"</p>
vt220	<p>From Wikipedia, the free encyclopedia:</p> <p>"DEC VT220 terminal with LK201 keyboard.</p>

TERM	DEFINITION
	<p>The VT220 was a terminal produced by Digital Equipment Corporation from 1983 to 1987.[1][2]</p> <p>Hardware</p> <p>The VT220 improved on the earlier VT100 series of terminals with a redesigned keyboard, much smaller physical packaging, and a much faster microprocessor. To meet the needs of various national regulatory agencies[citation needed], the VT220 was available with CRTs that used white, green, or amber phosphors.</p> <p>Several of the VT2xx models were pyramid shaped, allowing them to sit on a table top, leaving the surface of the display at an angle to the user; this angle could be adjusted. Because it was lower than head height, the result was an especially ergonomic terminal. The LK201 keyboard supplied with the VT220 was one of the first full length low profile keyboards available; it was developed at DEC's Roxbury, Massachusetts facility.</p> <p>The VT240 and VT241 were variants of the VT220, both capable of displaying vector graphics using the ReGIS instruction set. The VT241 was equipped with a color screen. The successor of the VT220 was the VT320, itself followed by the VT420.</p> <p>DEC VT220 connected to the serial port of a modern computer.</p> <p>Software</p> <p>The VT220 was designed to be compatible with the VT100, but added features to make it more suitable for an international market. This was accomplished with the National Replacement Character Set feature (e.g., Multinational Character Set) and support for 8-bit downloadable character sets."</p>
wxEmbedded	<p>From http://www.koansoftware.com/en/content/wxembedded :</p> <p>"wxEmbedded</p> <p>What is wxEmbedded®</p> <p>wxEmbedded® name and logo are registered trademarks of KOAN Software</p> <p>wxEmbedded - Embedded crossplatform GUI Library</p> <p>On March 2002 Koan software announced that a new project has been started by our company with wx-developers group.</p> <p>"After years of wishing, several recent projects have brought this closer to being a reality thanks to KOAN who has given the motivation behind all wxEmbedded project"</p> <p>-- Julian Smart, wxWidgets founder</p> <p>Here are the current strands in the wxEmbedded strategy, some points are already working, some are works in progress</p> <ul style="list-style-type: none"> ▪ wxWidgets for X11 (wxX11) ▪ wxWidgets for GTK+ (wxGTK) ▪ wxWidgets for Nano-X (wxNano-X) ▪ wxWidgets for Microwindows (wxMicrowindows) ▪ wxWidgets for SciTech MGL (wxMGL) ▪ wxWidgets for MS Windows CE (wxWinCE) ▪ Host tools, such as wxEmulator <p>Platforms supported are : x86 and ARM"</p>

TERM	DEFINITION
wxPython	A popular Python language binding for the cross-platform, "wxWidgets" GUI Toolkit.
wxWidgets	<p>A C++ library that lets developers create applications for Windows, OS X, Linux and UNIX on 32-bit and 64-bit architectures as well as several mobile platforms including Windows Mobile, iPhone SDK and embedded GTK+.</p> <p>It has popular language bindings for Python, Perl, Ruby and many other languages.</p> <p>"wxWidgets" (formerly "wxWindows") is a widget toolkit for creating graphical user interfaces (GUIs) for cross-platform applications. wxWidgets enables a program's GUI code to compile and run on several computer platforms with minimal or no code changes. It covers systems such as Microsoft Windows, Mac OS X (Carbon and Cocoa), iOS (Cocoa Touch), GNU/Linux Linux/Unix (X11, Motif, and GTK+), OpenVMS, OS/2 and AmigaOS. A version for embedded systems is under development.</p>
wxWindows License	<p>From Wikipedia, the free encyclopedia</p> <p>'wxWidgets is distributed under a custom made wxWindows License, similar to the GNU Lesser General Public License, with an exception stating that derived works in binary form may be distributed on the user's own terms.[5] This license is a free software license approved by the FSF,[17] making wxWidgets free software. It has been approved by the Open Source Initiative (OSI).[18]"</p>
xterm	<p>From Wikipedia, the free encyclopedia:</p> <p>"Not to be confused with X terminal display hardware.</p> <p>In computing, xterm is the standard terminal emulator for the X Window System. A user can have many different invocations of xterm running at once on the same display, each of which provides independent input/output for the process running in it (normally the process is a Unix shell).</p> <p>xterm originated prior to the X Window System. It was originally written as a stand-alone terminal emulator for the VAXStation 100 (VS100) by Mark Vandevoorde, a student of Jim Gettys, in the summer of 1984, when work on X started. It rapidly became clear that it would be more useful as part of X than as a standalone program, so it was retargeted to X. As Gettys tells the story, "part of why xterm's internals are so horrifying is that it was originally intended that a single process be able to drive multiple VS100 displays." [2]</p> <p>After many years as part of the X reference implementation, around 1996 the main line of development then shifted to XFree86 (which itself forked from X11R6.3), and it is presently actively maintained by Thomas Dickey.</p> <p>Many xterm variants are also available.[3] Most terminal emulators for X started as variations on xterm."</p> <p>NOTES:</p> <p>The "tsWxGTUI_PyVx" Toolkit supports the xterm, xterm-color, xterm-16color and xterm-256color terminal emulators with the following limitations:</p> <ul style="list-style-type: none"> ▪ The "tsWxGTUI_PyVx" Toolkit supports the xterm, xterm-color and xterm-16color terminal emulators. The inability to change the curses palette made it necessary to map the 68-color wxPython palette into the default 8-color curses palette. ▪ The "tsWxGTUI_PyVx" Toolkit does NOT yet support the xterm-256color terminal emulator. The inability to recreate the 68-color wxPython palette results in inappropriate colors and the appearance of spurious lines streaking across the display.

TERM	TEST TYPE DEFINITION
Acceptance Testing	Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html Testing the system with the intent of confirming readiness of the product and customer acceptance.
Ad Hoc Testing	Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html Testing without a formal test plan or outside of a test plan. With some projects this type of testing is carried out as an adjunct to formal testing. If carried out by a skilled tester, it can often find problems that are not caught in regular testing. Sometimes, if testing occurs very late in the development cycle, this will be the only kind of testing that can be performed. Sometimes ad hoc testing is referred to as exploratory testing.
Alpha Testing	Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html Testing after code is mostly complete or contains most of the functionality and prior to users being involved. Sometimes a select group of users are involved. More often this testing will be performed in-house or by an outside testing firm in close cooperation with the software engineering department.
Automated Testing	Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html Software testing that utilizes a variety of tools to automate the testing process and when the importance of having a person manually testing is diminished. Automated testing still requires a skilled quality assurance professional with knowledge of the automation tool and the software being tested to set up the tests.
Beta Testing	Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html Testing after the product is code complete. Betas are often widely distributed or even distributed to the public at large in hopes that they will buy the final product when it is released.
Black Box Testing	Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html Testing software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as a specification or requirements document..
Compatibility Testing	Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html Testing used to determine whether other system software components such as browsers, utilities, and competing software will conflict with the software being tested.
Configuration Testing	Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html Testing to determine how well the product works with a broad range of hardware/peripheral equipment configurations as well as on different operating systems and software.
Functional Test	Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html Functional test can be defined as testing two or more modules together with the intent of finding defects, demonstrating that defects are not present, verifying that the module performs its intended functions as stated in the specification and establishing confidence that a program does what it is supposed to do.

TERM	TEST TYPE DEFINITION
Independent Verification and Validation (IV&V)	<p>Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html</p> <p>The process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and doesn't fail in an unacceptable manner. The individual or group doing this work is not part of the group or organization that developed the software. A term often applied to government work or where the government regulates the products, as in medical devices.</p>
Installation Testing	<p>Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html</p> <p>Testing with the intent of determining if the product will install on a variety of platforms and how easily it installs.</p>
Integration Testing	<p>Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html</p> <p>Testing two or more modules or functions together with the intent of finding interface defects between the modules or functions. Testing completed at as a part of unit or functional testing, and sometimes, becomes its own standalone test phase. On a larger level, integration testing can involve a putting together of groups of modules and functions with the goal of completing and verifying that the system meets the system requirements. (see system testing)</p>
Load Testing	<p>Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html</p> <p>Testing with the intent of determining how well the product handles competition for system resources. The competition may come in the form of network traffic, CPU utilization or memory allocation.</p>
Performance Testing	<p>Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html</p> <p>Testing with the intent of determining how quickly a product handles a variety of events. Automated test tools geared specifically to test and fine-tune performance are used most often for this type of testing.</p>
Pilot Testing	<p>Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html</p> <p>Testing that involves the users just before actual release to ensure that users become familiar with the release contents and ultimately accept it. Often is considered a Move-to-Production activity for ERP releases or a beta test for commercial products. Typically involves many users, is conducted over a short period of time and is tightly controlled. (see beta testing)</p>
Regression Testing	<p>Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html</p> <p>Testing with the intent of determining if bug fixes have been successful and have not created any new problems. Also, this type of testing is done to ensure that no degradation of baseline functionality has occurred.</p>
Security Testing	<p>Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html</p> <p>Testing of database and network software in order to keep company data and resources secure from mistaken/accidental users, hackers, and other malevolent attackers.</p>
Software Testing	<p>Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html</p> <p>The process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and doesn't fail in an unacceptable manner. The organization and management of individuals or groups doing this work is not relevant. This term is often applied to commercial products such as internet applications. (contrast with independent verification and validation)</p>

TERM	TEST TYPE DEFINITION
Stress Testing	<p>Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html</p> <p>Testing with the intent of determining how well a product performs when a load is placed on the system resources that nears and then exceeds capacity.</p>
System Integration Testing	<p>Testing a specific hardware/software installation. This is typically performed on a COTS (commercial off the shelf) system or any other system comprised of disparate parts where custom configurations and/or unique installations are the norm.</p>
System Test	<p>Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html</p> <p>Several modules constitute a project. If the project is long-term project, several developers write the modules. Once all the modules are integrated, several errors may arise. The testing done at this stage is called system test.</p> <p>System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.</p> <p>Testing a specific hardware/software installation. This is typically performed on a COTS (commercial off the shelf) system or any other system comprised of disparate parts where custom configurations and/or unique installations are the norm.</p>
Unit Test	<p>Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html</p> <p>The first test in the development process is the unit test. The source code is normally divided into modules, which in turn are divided into smaller units called units. These units have specific behavior. The test done on these units of code is called unit test. Unit test depends upon the language on which the project is developed. Unit tests ensure that each unique path of the project performs accurately to the documented specifications and contains clearly defined inputs and expected results.</p>
Unit Testing	<p>Testing of individual hardware or software units or groups of related units</p> <p>-- <i>Institute of Electrical and Electronics Engineers. IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. New York, NY: 1990.</i></p>
Unit Testing (Unit)	<p>A "unit" (as defined in the software testing literature) is the smallest piece of code software that can be tested in isolation.</p> <ol style="list-style-type: none"> 1 "In isolation" means separate and apart from the application, and all other units. 2 The usual connotations associated with a unit are that <ol style="list-style-type: none"> a) It is a compilation unit recognized by the compiler b) It occupies a single file c) It is small (in "lines of code" sense) d) It is atomic, i.e., you normally do not think of breaking the unit into smaller pieces 3 Units, in object-oriented software, are classes, metaclasses, parameterized classes, and their corresponding instances. In programming languages that allow them, stand-alone procedures and functions are also units.

TERM	TEST TYPE DEFINITION
Unit Testing (Isolation)	<p>Unit testing (as defined in the software testing literature) requires that we test the unit in isolation. That is, we want to be able to say, to a very high degree of confidence, that any actual results obtained from the execution of test cases are purely the result of the unit under test. The introduction of other units may color our results.</p> <p>Since we can seldom, if ever, test a unit without the presence of at least some other software, unit testing involves such things as "drivers" and "stubs."</p> <p>Traditionally, a driver is a piece of software that controls (drives) the unit being tested. Drivers are usually thought of as invoking, or at least containing, the unit being tested, i.e., units (being tested) are subordinate to their respective drivers.</p> <p>Sometimes, units require the presence of other subordinate (to the unit) software. Traditionally, a stub is a piece of software that both mimics the characteristics of, and is (hopefully much) simpler than, a necessary piece of software that is immediately subordinate to the unit being tested.</p>
Unit Testing (Driver)	<p>Drivers are programs or tools that allow a tester to exercise/ examine in a controlling manner the unit of software being tested. A driver is usually expected to provide the following:</p> <ul style="list-style-type: none"> ▪ a means of defining, declaring, or otherwise creating, any variables, constants, or other items needed in the testing of the unit, and a means of monitoring the states of these items, ▪ any input and output mechanisms needed in the testing of the unit, ▪ a means of controlling the unit being tested, e.g., deciding when the unit will be invoked, and what information will be supplied upon invocation, ▪ the generation and/or handling of any necessary interrupts, and ▪ the generation and/or handling of any necessary exceptions.
Unit Testing (Stubs)	<p>Stubs are program units that are stand-ins for the other (more complex) program units that are directly referenced by the unit being tested. Stubs are usually expected to provide the following:</p> <ul style="list-style-type: none"> ▪ an interface that is identical to the interface that will be provided by the actual program unit, and ▪ the minimum acceptable behavior expected of the actual program unit. (This can be as simple as a return statement.) <p>The goal is to keep both drivers and stubs at a minimum level of complexity. If a driver or stub becomes too complex:</p> <ul style="list-style-type: none"> ▪ it will have to be formally tested itself, and ▪ the risk of the driver or stub masking, or contributing to, an error increases to an unacceptable level. <p>The simplest driver is indeed more complex than the simplest stub. However, drivers tend to reach a (manageable) maximum level of complexity and remain there, whereas it is not uncommon to see the complexity of (at least some) stubs come very close to the complexity of the units that they are supposed to be representing.</p> <p>Lastly, there are tools (e.g., test bed generators and test harnesses) that can automate the generation of drivers and stubs.</p>
User Acceptance Testing	<p>Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html</p> <p>See Acceptance Testing.</p>

TERM	TEST TYPE DEFINITION
White Box Testing	Excerpt From: http://www.exforsys.com/tutorials/testing/testing-types.html Testing in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose.

Draft

3 SYSTEM REQUIREMENTS

System Administrators, Software Engineers and System Operators use the following hardware and software platform component configurations:

- *Platform Configurations (Release Notes)* (on page 55)
- *Network Configuration (Release Notes)* (see "*Network Configurations (Release Notes)*" on page 65)
- *Development & Test Platforms (Release Notes)* (on page 73)

3.1 Platform Configurations (Release Notes)

System Administrators, Software Engineers, System Operators and Field Service Personnel may use a broad range of platform hardware and software configurations. Candidate configurations include or are equivalent to the following:

Operating System Software	Add-On Software	Central Processing Unit Hardware
GNU/Linux (CentOS 7.0-7.2, Debian 8/8.5.0, Fedora 21-25, OpenSUSE 13.1-13.2, Scientific 6.5-7.2, Ubuntu 12.04, 14.04, 15.04 and 16.04 etc.)	<ul style="list-style-type: none"> ▪ Python 2.6.x ▪ Python 2.7.x ▪ Python 3.2.x ▪ Python 3.3.x ▪ Python 3.4.x ▪ Python 3.5.x ▪ Python 3.6.x 	<ul style="list-style-type: none"> ▪ Intel & AMD / PC Compatible (32-bit & 64-bit) ▪ Freescale / IBM / Motorola PowerPC Compatible (32-bit & 64-bit) ▪ IBM Cell Broadband Engine (64-bit)
Mac OS X (10.3.x Panther - 10.12.x Sierra etc.)	<ul style="list-style-type: none"> ▪ Python 2.6.x ▪ Python 2.7.x ▪ Python 3.2.x ▪ Python 3.3.x ▪ Python 3.4.x ▪ Python 3.5.x ▪ Python 3.6.x ▪ Parallels Desktop 5 / 6 / 7 / 8 / 9 / 10 / 11 / 12 for Mac (runs various Linux, Unix and Windows Virtual Machines on Mac OS X using shared 	<ul style="list-style-type: none"> ▪ Intel & AMD / PC Compatible (32-bit & 64-bit) ▪ Freescale / IBM / Motorola PowerPC Compatible (32-bit & 64-bit)

	<p>computer processor, memory, peripheral, and network resources)</p> <ul style="list-style-type: none"> ▪ VMware Fusion 3 / 4 / 5 / 7 (runs various Linux, Unix and Windows Virtual Machines on Mac OS X using shared computer processor, memory, peripheral, and network resources) 	
<p>Microsoft Windows (XP, Vista, 7, 8, 8.1, 10 etc. which require Cygwin 1.7.x-2.5.x, the free and open-source, GNU/Linux-like command line interface and GNU tool-chain plug-in from Red Hat.)</p> <p>Cygwin consists of two parts:</p> <ul style="list-style-type: none"> ▪ a dynamic-link library (DLL) as an API compatibility layer providing a substantial part of the POSIX API functionality, and ▪ an extensive collection of software tools and applications that provide a Unix-like look and feel. 	<ul style="list-style-type: none"> ▪ Python 2.6.x ▪ Python 2.7.x ▪ Python 3.2.x ▪ Python 3.3.x ▪ Python 3.4.x ▪ Python 3.5.x ▪ Python 3.6.x 	<ul style="list-style-type: none"> ▪ Intel & AMD / PC Compatible (32-bit & 64-bit)
<p>UNIX (FreeBSD 9.2-10 Server, PC-BSD 10 Desktop, OpenIndiana 151a8, Solaris 11, SunOS 5.11 etc.)</p>	<ul style="list-style-type: none"> ▪ Python 2.6.x ▪ Python 2.7.x ▪ Python 3.2.x ▪ Python 3.3.x ▪ Python 3.4.x ▪ Python 3.5.x ▪ Python 3.6.x 	<ul style="list-style-type: none"> ▪ Intel & AMD / PC Compatible (32-bit & 64-bit) ▪ Freescale PowerPC Compatible (32-bit & 64-bit) ▪ Oracle/Sun SPARC or Compatible (32-bit & 64-bit) ▪ Silicon Graphics Incorporated MIPS or Compatible (32-bit & 64-bit)

See also: **Appendix A - Baseline Host Computer Platforms** (on page 239). It describes the development and test platforms used by the *TeamSTARS* "tsWxGTUI_PyVx" Toolkit author(s).

Having access to engineering documentation and source code empowers the "tsWxGTUI_PyVx" Toolkit recipients to "port" the Toolkit to other platforms and Python versions by:

- 1) Resolving some compatibility issues simply by modifying its source code to apply Python's "`__future__`" module feature.
- 2) Resolving remaining compatibility issues by the non-trivial task of re-engineering various *TeamSTARS* "tsWxGTUI_PyVx" Toolkit components.

3.1.1 Currently Used Platforms (Release Notes)

Host Computer Platform Configurations currently used by "tsWxGTUI_PyVx" Toolkit developers:

Draft

Draft

Make & Model	Hardware	Software
Apple 27" iMac Desktop	<p>2013-model year, 3.5 GHz Intel Quad Core i7 processor-based desktop with 16 GB RAM, 2560x1440 pixel resolution display and sufficient performance, resources and expansion capabilities to serve as the high-level baseline development and operator platform.</p> <ul style="list-style-type: none"> ▪ Its 3 TB (7200 RPM) SATA 6 Gb/s internal hard drive had 128 GB Solid State Flash memory and was used to store and run Apple's Mac OS X and the hypervisor virtualization applications (Parallels Desktop for Mac and VMware Fusion for Mac) that supported various guest operating systems. ▪ Hewlett-Packard Company P2015DN Series (26A5C8) LaserJet Printer connected via Ethernet ▪ Hewlett-Packard Company OfficeJet Pro 8500A (A910) e-All-in-One Series Printer connected via Wi-Fi or USB. 	<p>Host Operating System</p> <ul style="list-style-type: none"> ▪ Apple's Mac OS X (initially Mavericks releases 10.9.x and currently Sierra releases 10.12.x) with Python 2.7.10, Python 3.5.0 and Python 3.6.0 <p>Host Hypervisor Virtualization Applications</p> <ul style="list-style-type: none"> ▪ Parallels Desktop 9-12 for Mac ▪ VMware Fusion 5 & 7.1.3 for Mac <p>Concurrent or Interchangeable Guest Operating Systems (cloned from Apple 17" MacBook Pro Laptop and configured to share host drives and use 1440 x 900 display)</p> <ul style="list-style-type: none"> ▪ eComStation Demo CD version 2.2 BETA (IBM OS/2 4.5 Warp descendant) runs only what was shipped on the live CD and does not include Python. Most recent Python port for OS/2 & eComStations is Python 2.7.5. For details, see the following: "http://os2ports.smedley.id.au/index.php?page=python" and "http://blog.python.org/2011/05/python-33-to-drop-support-for-os2.html" ▪ GNU/Linux (CentOS 7.2 64-bit, Debian 8.5.0 64-bit, Fedora 22-24 64-bit, LXLE 14.02 32-bit, OpenSUSE 12.2 & 13.2 64-bit, Scientific 6.4 & 7.2 64-bit, Ubuntu 12.04, 14.04, 15.05 & 16.04 32-/64-bit with Python 2.7.x and 3.4.x/3.5.x. ▪ Microsoft Windows (10.0 Professional 64-bit, 10.0 Professional 32-bit, 8.1 Professional 32-bit, 8 Professional 32-bit, 7 Professional 32-bit with SP1, XP Professional 32-bit with SP3 and 2000 Professional 32-bit with SP2) with POSIX-like Cygwin plug-in and Python 2.6, 2.7, 3.2, 3.3 and 3.4. (NOTE: Windows 2000 came with obsolete Web Browser that precluded Windows 2000 updates and installation of Cygwin. After copying Windows 2000 updates and Cygwin directory from XP, Windows 2000 ran the TeamSTARS "tsWxGTUI_PyVx" Toolkit CLI and GUI tests but did not support mouse even with xterm.) ▪ UNIX (PC-BSD 9.2 & 10.3 64-bit without Parallels Tools, OpenIndiana 151a8 32-bit without Parallels Tools, a replacement for unreleased OpenSolaris 11/SunOS 5.11)

		with Python 2.6.4 running on Apple MacBook Pro
Apple 17" MacBook Pro Laptop	<p>2007-model year, 2.33 GHz Intel Core 2 Duo processor-based laptop with 4 GB RAM, 1920x1200 pixel resolution display and sufficient performance, resources and expansion capabilities to serve as the moderate-level baseline development and operator platform.</p> <ul style="list-style-type: none"> Its 160 GB (5400 RPM) SATA 1.5 Gb/s internal hard drive was used to run Apple's Mac OS X 10.7 and the hypervisor virtualization applications (Parallels Desktop 8 and VMware Fusion 5) that supported various guest operating systems. Its 1.5 TB (7200 RPM) SATA 3 Gb/s external hard drive was used to store and run configured versions of the Ubuntu Linux (12.04), Microsoft Windows (8 Pro, 7 Pro and XP Pro) and Unix (OpenSolaris 11/OpenIndiana 151) guest operating systems. Hewlett-Packard Company P2015DN Series (26A5C8) LaserJet Printer connected via Ethernet Hewlett-Packard Company OfficeJet Pro 8500A (A910) e-All-in-One Series Printer connected via Wi-Fi or USB. 	<p>Host Operating System</p> <ul style="list-style-type: none"> Apple's Mac OS X 10.7.5 with Python 2.6.8, 2.7.5, 3.2.2 <p>Host Hypervisor Virtualization Applications</p> <ul style="list-style-type: none"> Parallels Desktop 8 VMware Fusion 5 <p>Interchangeable Guest Operating Systems (configured to share host drives and use 1440 x 900 display)</p> <ul style="list-style-type: none"> Linux (Fedora 20 64-bit, OpenSUSE 12.2 64-bit, Scientific (CentOS) 6.4-6.5 64-bit, Ubuntu 12.04 32-bit) with Python 2.7 and 3.2. Windows (8.1 Professional 32-bit, 8 Professional 32-bit, 7 Professional 32-bit with SP1, XP Professional 32-bit with SP3) with POSIX-like Cygwin plug-in and Python 2.6, 2.7, 3.2 and 3.3 UNIX (PC-BSD 9.2-10.0 64-bit without Parallels Tools, OpenIndiana 151A8 32-bit without Parallels Tools, a replacement for unreleased OpenSolaris 11/SunOS 5.11) with Python 2.6.4 running on Apple MacBook Pro
Dell 15.6" Inspiron 7000 Laptop	<p>1998-model year, 366 MHz Intel Pentium II-based laptop with 384 MB RAM, 640x480/1024x768 pixel resolution display and expansion capabilities (marginal resources and performance) sufficient enough to serve as the low-level baseline development and operator platform.</p> <ul style="list-style-type: none"> Its interchangeable 32 GB (4200 RPM) ATA hard drives were used to run Windows XP Pro or Ubuntu 12.04 Linux. The platform's limited memory and available PCMCIA network adapters were incompatible with later versions of Windows or with other Linux distributions. (Windows XP recognized Xircom Ethernet and 3Com Wireless adapters; Ubuntu Linux 12.04 recognized only Linksys Wireless adapter.) Hewlett-Packard Company Photosmart C3180 All-in-One Printer, Scanner, Copier connected via USB. 	<p>Interchangeable Host Operating System</p> <ul style="list-style-type: none"> Windows XP Professional with SP3) with POSIX-like Cygwin plug-in and Python 2.6, 2.7, 3.2 and 3.3 Linux (Ubuntu 12.04) with Python 2.7 and 3.3

NOTE: Since cross-platform operating system and Python virtual machine technology is also available for

non-Intel based systems, it is likely that the TeamSTARS "tsWxGTUI_PyVx" toolkit will also work on those systems which use the equivalent operating systems and Python virtual machines with 32-bit and 64-bit microprocessors from other manufacturers including:

- *AMD*
- *ARM Holdings*
- *Cyrix*
- *Freescape*
- *Intel*
- *IBM*
- *Marvell*
- *NexGen*
- *Nvidia Tegra*
- *Oracle (previously Sun)*
- *OWC*
- *Qualcomm*
- *Rise Technology*
- *Samsung*
- *SigmaTel*
- *Texas Instruments*
- *Transmeta*
- *tilera*
- *Via (Centaur Technology division)*
- *winchip*

3.1.2 Previously Used Platforms (Release Notes)

Host Computer Platform Configurations previously used by "tsWxGTUI_PyVx" Toolkit developers:

Make & Model	Hardware	Software
Apple 17" MacBook Pro Laptop	<p>2007-model year, 2.33 GHz Intel Core 2 Duo processor-based laptop with 4 GB RAM and sufficient performance, resources and expansion capabilities to serve as the moderate-level baseline development and operator platform.</p> <ul style="list-style-type: none"> Its 160 GB (5400 RPM) SATA 1.5 Gb/s internal hard drive was used to run Apple's Mac OS X 10.7 and the hypervisor virtualization applications (Parallels Desktop 4-7 and VMware Fusion 4-5) that supported various guest operating systems. Its 1.5 TB (7200 RPM) SATA 3 Gb/s external hard drive was used to store and run configured versions of the Ubuntu Linux (10.04), Microsoft Windows (7 Pro and XP Pro) and Unix (OpenSolaris 11/OpenIndiana 151) guest operating systems. 	<p>Host Operating System</p> <ul style="list-style-type: none"> Apple's Mac OS X 10.4-10.7 with Python 2.6.8 <p>Host Hypervisor Virtualization Applications</p> <ul style="list-style-type: none"> Parallels Desktop 4-7 VMware Fusion 4-5 <p>Interchangeable Guest Operating Systems (configured to share host drives and use 1440 x 900 display)</p> <ul style="list-style-type: none"> Linux (Ubuntu 10.04) with Python 2.7 and 3.2 Windows (7 Professional / XP Professional with SP3) with POSIX-like Cygwin plug-in and Python 2.6, 2.7 and 3.2 UNIX (OpenIndiana 151a5, a replacement for unreleased OpenSolaris 11/SunOS 5.11) with Python 2.6.4 running on Apple MacBook Pro
Dell 15.6" Inspiron 7000 Laptop	<p>1998-model year, 366 MHz Intel Pentium II-based laptop with 384 MB RAM and expansion capabilities (marginal resources and performance) sufficient enough to serve as the low-level baseline development and operator platform.</p> <ul style="list-style-type: none"> Its interchangeable 32 GB (4200 RPM) ATA hard drives were used to run Windows XP Pro or Ubuntu 12.04 Linux. The platform's limited memory and available PCMCIA network adapters were incompatible with later versions of Windows or with other Linux distributions. (Windows XP recognized Xircom Ethernet and 3Com Wireless adapters; Ubuntu Linux 12.04 recognized only Linksys Wireless adapter.) 	<p>Interchangeable Host Operating System</p> <ul style="list-style-type: none"> Windows XP Professional with SP3) with POSIX-like Cygwin plug-in and Python 2.6, 2.7 and 3.2 Linux (Ubuntu 12.04) with Python 2.7 and 3.2

<p>Notes - Baseline Host Computer Platform Configurations previously used by "tsWxGTUI_PyVx" Toolkit developers</p>	<ol style="list-style-type: none"> 1 Linux (Fedora 15-16) with Python 2.6, 2.7 and 3.2 running on Apple MacBook Pro - 32-/64-bit Intel Core 2 Duo Processors running with 4 GB RAM Linux Virtual Machine created and managed by Parallels Desktop 6 for Mac 2 Linux (OpenSUSE 11) with Python 2.6, 2.7 and 3.2 running on Apple MacBook Pro - 32-/64-bit Intel Core 2 Duo Processors with 4 GB RAM running Virtual Machine created and managed by Parallels Desktop 6 for Mac 3 Linux (Ubuntu 10.04) with Python 2.7 and 3.2 running on Apple MacBook Pro - 32-/64-bit Intel Core 2 Duo Processors with 4 GB RAM running Linux Virtual Machine created and managed by Parallels Desktop 6 for Mac 4 Mac OS X (Tiger 10.4.0-Snow Leopard 10.6.8) with Python 2.6, 2.7 and 3.2 running on Apple MacBook Pro - 32-/64-bit Intel Core 2 Duo Processors with 4 GB RAM running Mac OS 10.4-10.6. 5 Windows (XP-SP3) with UNIX-like Cygwin plug-in and Python 2.6 running on Dell Laptop - 32-bit Intel Pentium 2 Processor with 384 MB RAM running Windows XP-SP3 6 Windows (XP-SP3 and Release 8 Preview) with UNIX-like Cygwin plug-in and Python 2.6 running on Apple MacBook Pro - 32-/64-bit Intel Core 2 Duo Processors with 4 GB RAM running Windows Virtual Machine created and managed by Parallels Desktop 6 for Mac
---	--

Notes - Experimental Host Computer Platform Configurations previously used by "tsWxGTUI_PyVx" Toolkit developers	<p>1 Windows (7 Professional) with built-in "Command Prompt" accessory shell and Python 2.7 running on Apple MacBook Pro - 32-/64-bit Intel Core 2 Duo Processors with 4 GB RAM running Windows Virtual Machine created and managed by Parallels Desktop (8) for Mac.</p>
	<p>This configuration uses the Windows built-in "Command Prompt" accessory shell which can run Command Line Interface components ("tsLibCLI", "tsTestsCLI" and "tsToolsCLI") of the "tsWxGTUI_PyVx" toolkit.</p> <p>It does NOT support the low-level, "nCurses" library, a pre-requisite for running the Graphical-style User Interface components ("tsLibGUI", "tsTestsGUI" and "tsToolsGUI") of the "tsWxGTUI_PyVx" toolkit.</p>
	<p>2 Windows (7 Professional) with UNIX-like Git Bash plug-in and Python 2.7 running on Apple MacBook Pro - 32-/64-bit Intel Core 2 Duo Processors with 4 GB RAM running Windows Virtual Machine created and managed by Parallels Desktop (8) for Mac.</p>
	<p>This configuration, like those using Cygwin, includes a Bash shell which can run Command Line Interface components ("tsLibCLI", "tsTestsCLI" and "tsToolsCLI") of the "tsWxGTUI_PyVx" toolkit.</p> <p>Unlike those configurations using Cygwin, it does NOT support the low-level, "nCurses" library, a pre-requisite for running the Graphical-style User Interface components ("tsLibGUI", "tsTestsGUI" and "tsToolsGUI") of the "tsWxGTUI_PyVx" toolkit.</p>
	<p>3 Windows (7 Professional) and Python 2.7 with the GNUwin32 and PDCurses Version 2.6 plug-ins running on Apple MacBook Pro - 32-/64-bit Intel Core 2 Duo Processors with 4 GB RAM running Windows Virtual Machine created and managed by Parallels Desktop (8) for Mac.</p>
	<p>This configuration, unlike those using Cygwin, includes a DOS-like Command Prompt shell which can run Command Line Interface components ("tsLibCLI", "tsTestsCLI" and "tsToolsCLI") of the "tsWxGTUI_PyVx" toolkit.</p> <p>Like those configurations using Cygwin, PDCurses Version 2.6 does support the low-level, Python "Curses" library, a pre-requisite for running the Graphical-style User Interface components ("tsLibGUI", "tsTestsGUI" and "tsToolsGUI") of the "tsWxGTUI_PyVx" toolkit. It runs the test_PDCurses (renamed test_tsWxCurses) application. However, PDCurses Version 2.6 traps because it lacks the mouse button definitions needed by the "tsWxGraphicalTextUserInterface" module to emulate "wxPython" in order to run such applications as "test_tsWxWidgets.py".</p>

3.2 Network Configurations (Release Notes)

System Administrators, Software Engineers, System Operators and Field Service Personnel may use platform hardware and software in the following network configurations:

- ***Stand Alone System Architecture*** (on page 65) - Hardware and Software for a Single Host Computer platform.
- ***Stand Among System Architecture*** (on page 69) - Hardware and Software for a Single Host Computer connected, via a wired or wireless network and additional Hardware and Software, with one or more remote Host Computer platforms.

3.2.1 Stand Alone System Architecture

The *TeamSTARS* "tsWxGTUI_PyVx" Toolkit provides:

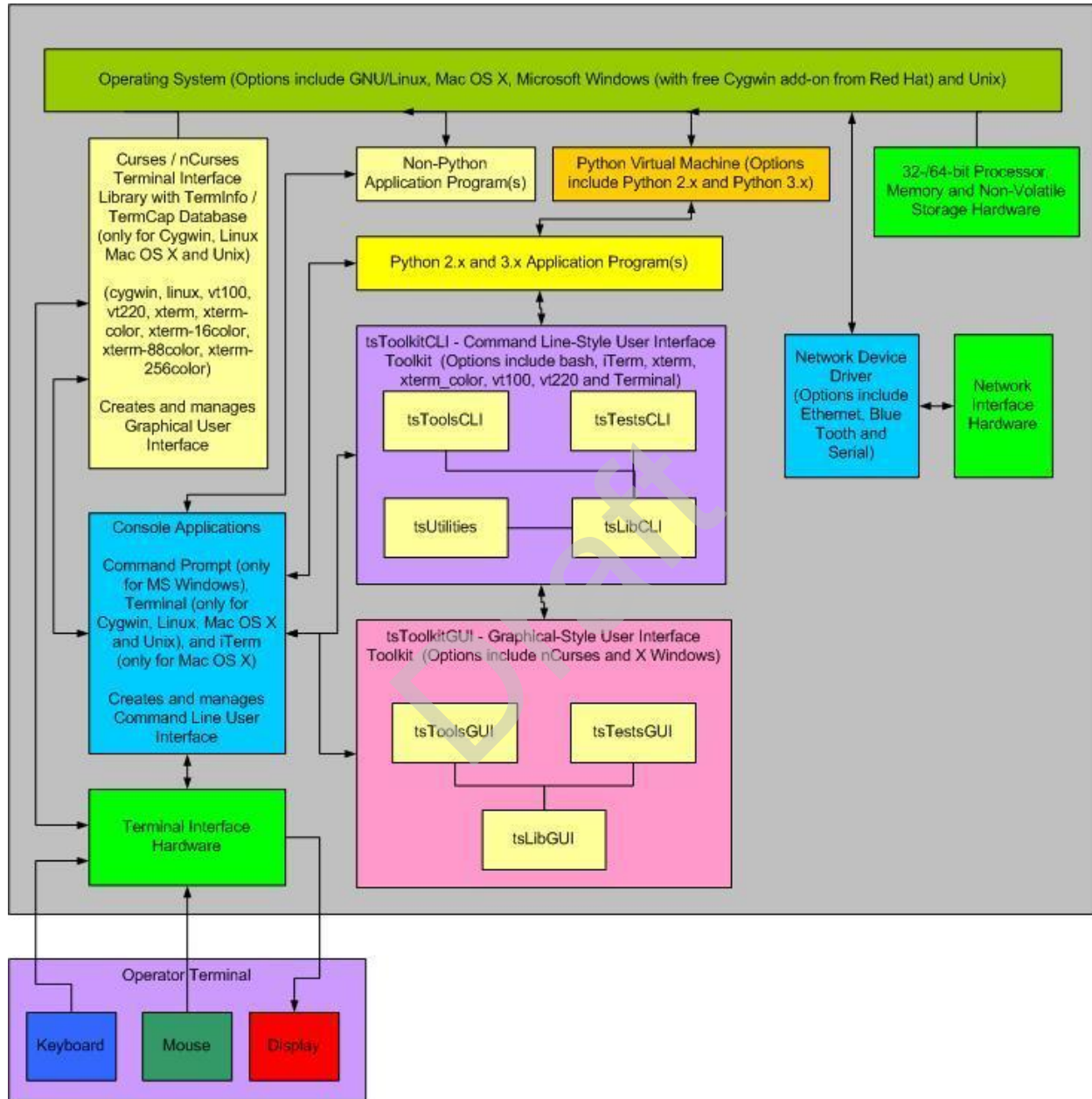
1. Python version-specific components for its Command Line Interface ("tsToolkitCLI")
2. Python version-specific components for its Graphical-style User Interface ("tsToolkitGUI").

The following description uses the component names as depicted in the Block Diagram

This section depicts and describes the organization, function of and interface between various system hardware and software components and "tsWxGTUI_PyVx" Toolkit users (System Administrator, Software Engineer, System Operator and Field Service personnel):

- 1** the external System Operator interface to "tsToolkitCLI"
- 2** the internal "tsToolkitCLI" interface to "tsToolkitGUI"
- 3** the internal System Operator interfaces:
 - a) to "tsLibCLI", "tsToolsCLI" . "tsTestsCLI" and "tsUtilities" via "tsToolkitCLI"
 - b) to "tsLibGUI", "tsToolsGUI" and "tsTestsGUI" via "tsToolkitGUI" and "tsToolkitCLI"

This depiction represents a typical Stand Alone System configuration. In this configuration, the optional Network Hardware Interface and its associated Network Device Driver Interface should not be used, even if present, in order to avoid activities that adversely impact system performance.



- 1 **Operating System** - The platform specific software (such as Linux, MacOS X, Microsoft Windows and Unix) that coordinates and manages the time-shared use of a platform's processor, memory, storage and input/output hardware resources by multiple application programs and their associated users/operators.
- 2 **Operator Terminal** - A device for human interaction that includes:
 - a) A Keyboard unit for text input

- b) A Mouse unit (mouse, trackball, trackpad or touchscreen with one or more physical or logical buttons) for selecting one of many displayed GUI objects to initiate an associated action.
- c) A Display unit (1-color "ON"/"OFF" or multi-color two-dimensional screen) for output of text and graphic-style, tiled and overlaid boxes.

3 Terminal Hardware Interface - The platform specific hardware with connections to the device units of the Operator Terminal.

- a) A PS/2 Port is a type of input port developed by IBM for connecting a mouse or keyboard to a Personal Computer. It supports a mini DIN plug containing just 6 pins.
- b) An RS-232C or RS-422 port for connecting a mouse for position and button-click input.
- c) A Universal Serial Bus (USB) port is an industry standard first developed in the mid-1990s that was designed to standardize the connection of computer peripherals (including keyboards, pointing devices, digital cameras, printers, portable media players, disk drives and network adapters) to personal computers, both to communicate and to supply electric power. It has effectively replaced a variety of earlier interfaces, such as serial and parallel ports, as well as separate power chargers for portable devices.

The USB 1.0 specification was introduced in January 1996. It defined data transfer rates of 1.5 Mbit/s "Low Speed" and 12 Mbit/s "Full Speed". The first widely used version of USB was 1.1 (now called "Full-Speed"), which was released in September 1998. Its 12 Mbit/s data rate was intended for higher-speed devices such as disk drives, and its lower 1.5 Mbit/s rate for low data rate devices such as joysticks.

The USB 2.0 (now called "Hi-Speed") specification was released in April 2000. It defined a higher data transfer rate, with the resulting specification achieving 480 Mbit/s, a 40-times increase over the original USB 1.1 specification.

The USB 3.0 specification (now called "SuperSpeed") was released in November 2008. It defined an even higher data transfer rate (up to 5 Gbit/s) and was backwards-compatible with USB 2.0. It added a new, higher speed bus called SuperSpeed in parallel with the USB 2.0 bus.

- d) A Video Adapter is a computer circuit card that provides digital-to-analog conversion, video RAM, and a video controller so that data can be sent to a computer's display. It typically adheres to the de facto standard, Video Graphics Array (VGA). VGA describes how data - essentially red, green, blue data streams - is passed between the computer and the display. It also describes the frame refresh rates in hertz. It also specifies the number and width of horizontal lines, which essentially amounts to specifying the resolution of the pixels that are created. VGA supports four different resolution settings and two related image refresh rates. The maximum VGA resolution setting produces a display that is 640 pixels wide by 480 pixels high. For a character font that is 8 pixels wide by 12 pixels high, the longest line of text will be 80 characters wide and there can be up to 40 lines of text displayed at any moment. Higher resolutions, such as SVGA, are supported by more advanced Video Adapters. The higher resolution settings typically require use of proportionally larger displays in order to maintain the size and legibility of the displayed text.

4 Terminal Device Driver - The platform specific software for transforming data (such as single button scan codes, multi-button flags and pointer position) to and from the platform independent formats (such as upper and lower case text, display screen column and row and displayed colors, fonts and special effects) used by the Command Line Interface and Graphical User Interface software.

- 5 Command Line-Style Interface ("tsToolKitCLI")** - The platform specific keywords arguments, positional arguments and their associated values and syntax of text used to request services from the Operating System and various Application Programs.
- a) **"tsLibCLI"** --- A library of command line building blocks that establishes the POSIX-/Unix-style, run time environment for pre-processing source files, launching application programs, handling events (registering events with date, time and event severity annotations) and configuring console terminal and file system input and output.
 - b) **"tsToolsCLI"** --- A set of command line interface application programs and utility scripts for: checking source code syntax and style via "plint"; generating Unix-style "man page" documentation from source code comments via "pydoc"; and installing, modifying for publication and tracking software development metrics.
 - c) **"tsTestsCLI"** --- A set of command line interface application programs and utility scripts for unit, integration and system level regression testing.
 - d) **"tsUtilities"** --- A library of computer system configuration, diagnostic, installation, maintenance and performance tool components for various host hardware and software platforms.
- 6 Graphical-Style User Interface ("tsToolKitGUI")** - The platform specific tiled, overlaid and click-to-select Frames, Dialogs, Pull-down Menus, Buttons, CheckBoxes, Radio Buttons, Scrollbars and associated keywords, values and syntax of text used to request services from the Operating System and various Application Programs.
- a) **"tsLibGUI"**--- A library of graphical-style user interface building blocks that establishes the emulated run time environment for the high-level, pixel-mode, "wxPython" GUI Toolkit via the low-level, character-mode, "nCurses" Text User Interface Toolkit.
 - b) **"tsToolsGUI"**--- A set of graphical-style user interface application programs for tracking software development metrics.
 - c) **"tsTestsGUI"** --- A set of graphical-style user interface application programs and command line interface utility scripts for unit, integration and system level regression testing.
- 7 Python Application Program** - The application specific program that performs its service when executed by the Python Virtual Machine.
- 8 Non-Python Application Program** - The application specific program that performs its service when its pre-compiled, platform specific machine code is executed. Typically, these services are used to analyze, edit, view, copy, move or delete those data and log files which are of interest or no longer needed.
- 9 Python Virtual Machine** - The platform specific program that loads, interprets and executes the platform independent source code of a Python language application program.
- 10 Processor, Memory, Storage and Communication Hardware** - Platform specific resources that are required by the Operating System and Application software.
- 11 Network Hardware Interface** - The optional platform specific ethernet, blue-tooth and RS-232 serial port hardware for physical connections between the local system and one or more remote systems. It may also include such external hardware as gateways, routers, network bridges, switches, hubs, and repeaters. It may also include hybrid network devices such as multilayer switches, protocol converters, bridge routers, proxy servers, firewalls, network address translators, multiplexers, network interface controllers, wireless network interface controllers, modems, ISDN terminal adapters, line drivers, wireless access points, networking cables and other related hardware.

- a) An RJ-45 Ethernet port connecting to a local or wide area network. This port is capable of conducting simultaneous (full-duplex,) two-directional input and output at speeds over 100 gigabits per second. This port can also provide the interface to an optional printer shared with other network users.
- b) An RS-232C or RS-422 port for connecting a modem. Depending on the application, modems could be operated, at speeds over 56 kilobits per second, in either of two modes. In half duplex mode, each side alternated its sending and receiving roles. In full-duplex mode, each side could simultaneously send and receive.

12 Network Device Driver Interface - The optional platform specific software whose layered protocol suite (such as TCP/IP) enables the concurrent sharing of the physical connection between the local system and one or more remote systems.

NOTE: Concurrent local and remote login sessions are a convenience that must be used with caution. Time critical, resource intensive activities ought to be performed individually or sequentially (but NOT concurrently) on a Stand Alone System.

3.2.2 Stand Alone System Architecture

The *TeamSTARS* "tsWxGTUI_PyVx" Toolkit provides:

1. Python version-specific components for its Command Line Interface ("tsToolkitCLI")
2. Python version-specific components for its Graphical-style User Interface ("tsToolkitGUI").

The following description uses the component names as depicted in the Block Diagram

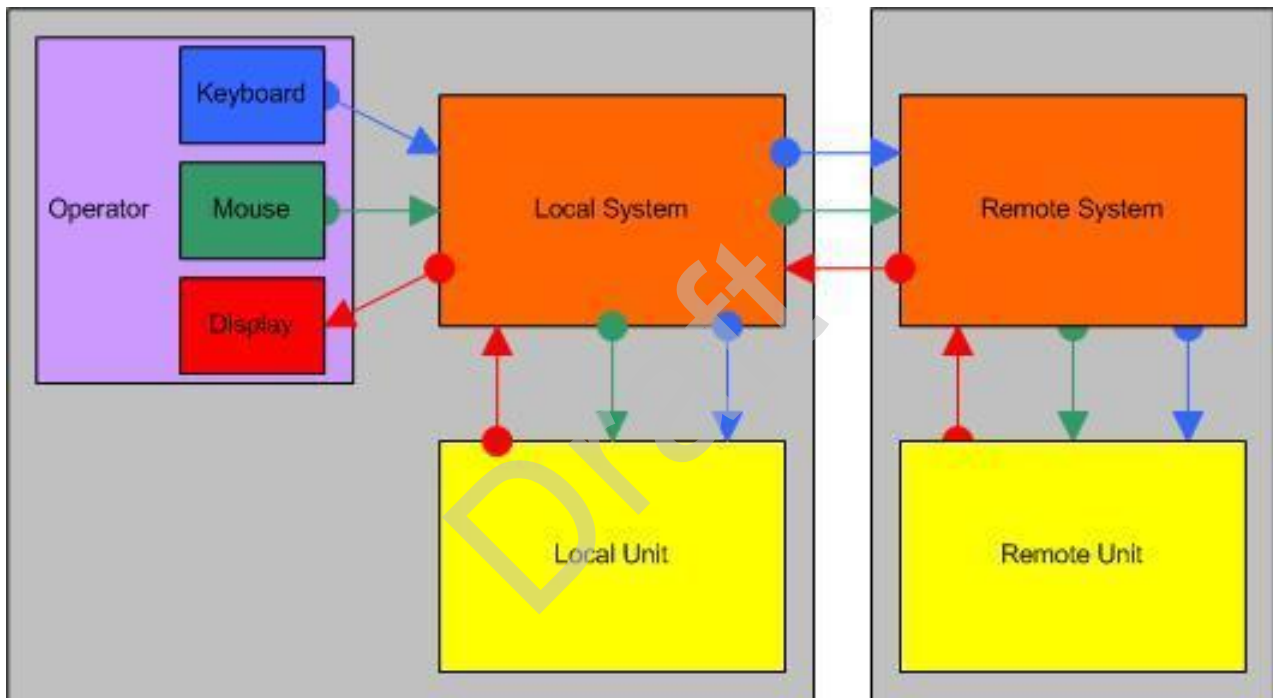
The ***Stand Alone System Architecture*** (on page 65), as previously described, may be extended to enable a single operator, working from a Local System, to interact with one or more Remote Systems.



In this configuration, the Local (Left) and Remote (Right) systems must first be networked via the available communication resources (Network Interface Hardware and Network Device Driver Interface).

Once networked, the local system operator must login to the Remote system via the "ssh user@Remote" command. The Local and Remote Terminal Device Interface then establishes a logical communication channel for exchanging keyboard, mouse and display information.

For each login Local and Remote session, the Operator may then select and run an Application Program. As depicted in the following figure. Application Programs run as Local Units on the system to which the Operator first logged in. Application Programs run as Remote Units on the systems to which the operator logged in via the "ssh user@Remote" command.



NOTE: Concurrent login sessions are a convenience that must be used with caution. Time critical, resource intensive activities ought to be performed individually or sequentially (but NOT concurrently) on a Stand Alone System. Only non-time critical, non-resource intensive activities may be performed concurrently on Stand Alone or Stand Among Systems.

- 1 **Local System (Left)** --- Operator opens one or more Command Line Interface Shells.
 - One or more newly opened shells may be used to run a Python or non-Python Application Program as its **Local Unit (Left.)**
 - One or more newly opened shells may be used to login to a Remote system (via the "ssh user@Remote" command). Once Remote login has been successful, the operator may run a Python or non-Python Application Program as a **Remote Unit (Right).**
- 2 **Local Unit (Left)** --- A Python or non-Python Application Program that has been launched in a Local Command Line Interface Shell.
- 3 **Remote System (Right)** - Same Operator opens one or more Command Line Interface Shells.

- One or more newly opened shells may be used to run a Python or non-Python Application Program as its **Remote Unit (Right)**.

The **Multi-Session Desktop** (on page 71) figure depicts the desktop of a multi-user, multi-process, multi-threaded computer running the Professional Edition of Microsoft Windows 7. Among the background of desktop icons, there are two *TeamSTARS* "tsWxGTUI_PyVx" Toolkit sessions. The time each session displays synchronizes within its own one second refresh interval. The local session, on the left, is actively running Python 3.x on the Windows platform. The remote session, on the right, is actively running Python 2.x on the Mac OS X Yosemite platform which also serves as the Parallels 10 Hypervisor host for diverse Guest Operating Systems including:

Linux (CentOS 7 64-bit, Fedora 21 64-bit, Scientific 6.5 32-bit and Ubuntu 12.04 32-bit)

Windows (98 SE 16-bit, XP 32-bit, 7 32-bit, 8 32-bit and 8.1 32-bit)

Unix (FreeBSD 9.2, PC-BSD 10, OpenIndiana 151a8 and OpenSolaris 11 32-bit)

- One or more newly opened shells may be used to login to a Remote system (via the "ssh user@Remote command). Once Remote login has been successful, the operator may run a Python or non-Python Application Program as a **Remote Unit (Right)**.

4 Remote Unit (Right) --- A Python or non-Python Application Program that has been launched in a Remote Command Line Interface Shell.

3.2.2.1 Multi-Session Desktop

From Wikipedia, the free encyclopedia

"In computer science, in particular networking, a session is a semi-permanent interactive information interchange, also known as a dialogue, a conversation or a meeting, between two or more communicating devices, or between a computer and user (see Login session). A session is set up or established at a certain point in time, and then torn down at some later point. An established communication session may involve more than one message in each direction. A session is typically, but not always, stateful, meaning that at least one of the communicating parts needs to save information about the session history in order to be able to communicate, as opposed to stateless communication, where the communication consists of independent requests with responses.

An established session is the basic requirement to perform a connection-oriented communication. A session also is the basic step to transmit in connectionless communication modes. However any unidirectional transmission does not define a session.[1]

Communication sessions may be implemented as part of protocols and services at the application layer, at the session layer or at the transport layer in the OSI model.

1 Application layer examples:

- a) HTTP sessions, which allow associating information with individual visitors
- b) A telnet remote login session

2 Session layer example:

- a) A Session Initiation Protocol (SIP) based Internet phone call

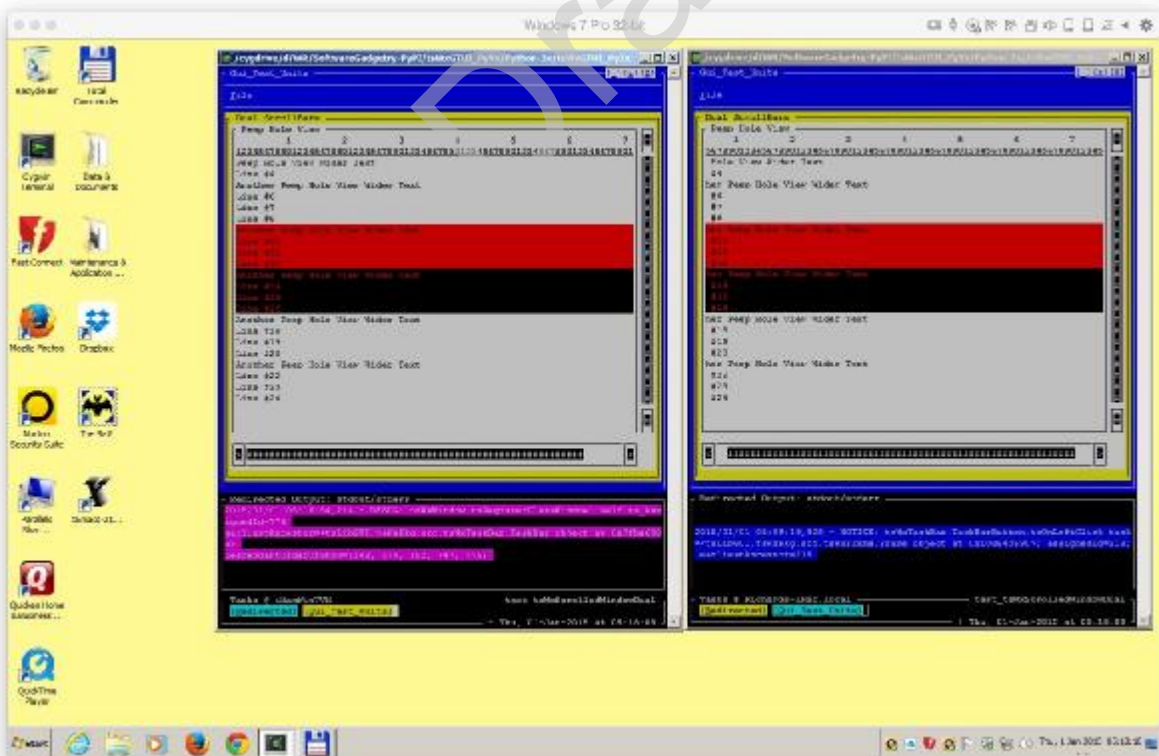
3 Transport layer example:

- a) A TCP session, which is synonymous to a TCP virtual circuit, a TCP connection, or an established TCP socket.

In the case of transport protocols that do not implement a formal session layer (e.g., UDP) or where sessions at the application layer are generally very short-lived (e.g., HTTP), sessions are maintained by a higher level program using a method defined in the data being exchanged. For example, an HTTP exchange between a browser and a remote host may include an HTTP cookie which identifies state, such as a unique session ID, information about the user's preferences or authorization level.

HTTP/1.0 was thought to only allow a single request and response during one Web/HTTP Session. However a workaround was created by David Hostettler Wain in 1996 such that it was possible to use session IDs to allow multiple phase Web Transaction Processing (TP) Systems (in ICL[disambiguation needed] nomenclature), with the first implementation being called Deity. Protocol version HTTP/1.1 further improved by completing the Common Gateway Interface (CGI) making it easier to maintain the Web Session and supporting HTTP cookies and file uploads.

Most client-server sessions are maintained by the transport layer - a single connection for a single session. However each transaction phase of a Web/HTTP session creates a separate connection. Maintaining session continuity between phases required a session ID. The session ID is embedded within the <A HREF> or <FORM> links of dynamic web pages so that it is passed back to the CGI. CGI then uses the session ID to ensure session continuity between transaction phases. One advantage of one connection-per-phase is that it works well over low bandwidth (modem) connections. Deity used a sessionID, screenID and actionID to simplify the design of multiple phase sessions."



The Sample Screenshot above shows a Windows 7 Desktop with:

- 1 A local Windows host with Python 3x session on left; and
- 2 A remote Mac OS X host with Python 2x session on right.
- 3 Each session consists of
 - a) a wxPython-style Frame named "Dual ScrollBars" containing scrollable text with optional color markup.
 - b) a wxPython-style Frame named "Redirected Output" containing date and time stamped event messages, with optional with color markup, that scroll up when new events are registered.
 - c) a Host Desktop-style Frame named "Tasks @ Host Name" and Application Name with buttons to shift focus from background to foreground. There is also a spinner (to indicate the frequency or absence of idle time) and the current date and time (to indicate when the display was last updated).

3.3 Development & Test Platforms (Release Notes)

See Microsoft Excel Spreadsheet file: "Platform_Configuration.xls".

Copies of the Spreadsheet are located in the following distribution documentation subdirectory:

- ["/Notebooks/EngineeringNotebook/MS-Excel-Files"]

The Spreadsheet contains the following worksheets:

- 1 "Host_Configurations"
 - a) **OS Type** (Linux, Mac OS X, Microsoft Windows, Unix etc.)
 - b) **Distribution** (Manufacturer, Product etc.)
 - c) **Datapath** (32-/64-bit)
 - d) **Release** (x.y.z code)
 - e) **Host Platform** (See Notes regarding representative 2013-model year, 2007-model year and 1999-model year platforms used for development and test)
 - f) **Python Release** (Capabilities and Limitations)
- 2 "Curses Terminal_Configuration"

"wxPython-style screen Size in pixels (character columns x rows)			
▪	wxSYS_SCREEN_NONE	= 0 #	not yet defined
▪	wxSYS_SCREEN_TINY	= 1 #	< 320x240 (40 col x 20 row)
▪	wxSYS_SCREEN_PDA	= 2 #	>= 320x240 (40 col x 20 row)
▪	wxSYS_SCREEN_SMALL	= 3 #	>= 640x480 (80 col x 40 row)
▪	wxSYS_SCREEN_DESKTOP	= 4 #	>= 800x600 (100 col x 50 row)
▪	cygwin console; 8-color without mouse		
▪	cygwin mintty; 16-color with multi-button/wheel, single-/double-/triple-click mouse		
▪	linux; 8-/16-color with multi-button/wheel, single-/double-/triple-click mouse		
▪	xterm; 8-color with multi-button/wheel, single-/double-/triple-click mouse		
▪	xterm_color; 8-color with multi-button/wheel, single-/double-/triple-click mouse		
▪	xterm-16color; 16-color with multi-button/wheel, single-/double-/triple-click mouse		
▪	xterm-88color; 88-color with multi-button/wheel, single-/double-/triple-click mouse		
▪	xterm-256color; 256-color with multi-button/wheel, single-/double-/triple-click mouse		
▪	vt100; non-color with or without limited function mouse (multi-button, single-click with slower response)		
▪	vt220; non-color with or without limited function mouse (multi-button, single-click with slower response)		

- a) **has_colors** (True or False)
- b) **curses_colors** (8, 16, 88 or 256)
- c) **curses_color_pairs** (64, 256, 7744 or 32768; typically `curses_colors**2`, unless constrained to lesser amount)

There is a control switch ("USE_256_COLOR_PAIR_LIMIT") in `./tsLibGUI/tsWxGlobals.py` which establishes 256 (16x16) as the maximum number of available color pairs (overriding the value reported by curses). As a consequence, the "tsWxGTUI" Toolkit will apply the xterm-16color palette instead of the xterm-88color and xterm-256color palette which cannot otherwise be made to work.

- d) **can_change_color** (True or False)
- e) **built-in** (default) palette (typically 8 or 16)
- f) **custom palette** (typically 8, 16, 71 or 140)

The "tsWxGTUI" Toolkit provides a mapping of the "wxPython"-style 71-colors to the available built-in 8-/16-color palette in order to fulfill its guarantee to recognize the "wxPython" standard 68-color palette.

4 PACKAGE CONTENTS

The "tsWxGTUI" Toolkit includes the following components:

- 1** Repository
- 2** Documents
- 3** ManPages
- 4** Notebooks
- 5** SourceDistributions
- 6** *Developer-Sandbox* (on page 105)
 - a) Command Line Interface Library and Tools
 - b) Graphical Text User Interface Library and Tools
- 7** *Site-Package* (on page 113)
 - a) Command Line Interface Library and Tools
 - b) Graphical Text User Interface Library and Tools

4.1 Repository

```
#-----
#"Time-stamp: <12/18/2016  2:42:42 PM rsg>
#-----

===== File: README2-Repository.txt =====

+-----+-----+ TeamSTARS "tsWxGTUI_PyVx" Toolkit
| ts | Wx |       with Python 2x & Python 3x based
+-----+-----+       Command Line Interface (CLI)
| G T U I |       and "Curses"-based "wxPython"-style,
+-----+-----+       Graphical-Text User Interface (GUI)

Get that cross-platform, pixel-mode "wxPython" feeling on
platforms with:

* 64-bit processors, nCurses 6.x, 64-bit Python 3.6.x or
  later GUI applications and character-mode 256-/16-/8-
  color (xterm-family) and non-color (vt100-family)
  terminals and terminal emulators.

* 32-bit processors, nCurses 6.x/5.x, 32-bit Python 3.5.2
  or earlier GUI applications and character-mode 16-/8-
  color (xterm-family) and non-color (vt100-family)
  terminals and terminal emulators.

You can find this and other plain-text files in the
Toolkit subdirectory named:

    "./<Toolkit Recipient's Repository>/Documents".

<Your Working Repository>
(e.g. "tsWxGTUI_PyVx_Repository")
|
|   Working repository containing directories and
|   files to be packaged into downloadable "tarball"
|   and/or "zip" files via the setup shell scripts
|   at the bottom of this diagram.
|
+-- ["Documents"] (Original)
|
|   |   This directory contains a collection of files
|   |   which provide the Toolkit recipient with an
|   |   understanding of the purpose, goals & capabil-
|   |   ities, non-goals & limitations, terms & condi-
|   |   tions and procedures for installing, operating,
|   |   modifying and redistributing the Toolkit.
|   |
|   +-- ["Announcement"]
|   :
|   :
|   +-- "README.txt"
```

```

|     +-- "README1-Introduction.txt"
|     +-- "README2-Repository.txt"
|     +-- "README3-Documents.txt"
|     +-- "README4-ManPages.txt"
|     +-- "README5-Notebooks.txt"
|     +-- "README6-SourceDistributions.txt"
|     +-- "README7-DeveloperSandboxes.txt"
|     +-- "README8-SitePackages.txt"
|     +-- "README9-KeyBoardMouseInput.txt"
|     +-- "GETTING_STARTED.txt"
|     +-- "INSTALL.txt"
|     +-- "DEMO.txt"
|     :
|     :
|     +-- "TROUBLESHOOT.txt"
|
+-- ["ManPages"] (Original)
|
|     Deliverable Toolkit manual pages are a
|     form of online software documentation
|     usually found on a Unix or Unix-like
|     operating system.
|
|     Topics covered include computer programs
|     (including library and system calls),
|     formal standards and conventions, and even
|     abstract concepts.
|
|     Unlike their Unix or Unix-like counterparts,
|     a Toolkit user may NOT invoke a man page by
|     issuing the "man command". Instead, a user
|     must display a man page by issuing the
|     "less <man document file>" command.
|
|     +-- ["tsManPagesLibCLI"]
|     +-- ["tsManPagesLibGUI"]
|     +-- ["tsManPagesTestsLibCLI"]
|     +-- ["tsManPagesTestsLibGUI"]
|     +-- ["tsManPagesToolsCLI"]
|     +-- ["tsManPagesToolsGUI"] (Future)
|     +-- ["tsManPagesToolsLibCLI"]
|     +-- ["tsManPagesToolsLibGUI"] (Future)
|     +-- ["tsManPagesUtilitiesCLI"] (Future)
|
+-- ["Notebooks"] (Original Pre-dates Documents)
|
|     Contains a collection of commentaries that
|     express opinions or offerings of explanations
|     about events or situations that might
|     be useful to Toolkit installers, developers,
|     operators, troubleshooters and distributors.
|     The documents may be in Application-specific
|     formats (such as Adobe PDF, JPEG Bit-mapped
|     image, LibreOffice, Microsoft Office, plain

```

```
|      | text) .  
|      |  
|      | +-- ["DeveloperNotebook"] (Future Original  
|      | | Developer-Sandbox)  
|      | |  
|      | | Contains a collection of:  
|      | | API-References-Pixel-Mode-wxPython  
|      | | and Developer-ReadMe-Files  
|      | |  
|      | +-- "README5-DeveloperNotebook.txt"  
|      |  
|      | +-- ["EngineeringNotebook"] (Future Original  
|      | | Developer-Sandbox)  
|      | |  
|      | | Contains a Toolkit Developer oriented collection of:  
|      | |  
|      | | Project (purpose,  
|      | | | goals,  
|      | | | non-goals,  
|      | | | features,  
|      | | | capabilities,  
|      | | | limitations),  
|      | |  
|      | | Plan (software life-cycle),  
|      | |  
|      | | Requirements (purpose,  
|      | | | goals,  
|      | | | non-goals,  
|      | | | features,  
|      | | | capabilities,  
|      | | | limitations,  
|      | | | file system configuration,  
|      | | | hardware & software interface,  
|      | | | software,  
|      | | | system,  
|      | | | user configuration options),  
|      | |  
|      | | Design (API emulation strategy, architecture),  
|      | |  
|      | | Implementation (developer-sandbox, site-package),  
|      | |  
|      | | Test (unit, integration, system, acceptance),  
|      | |  
|      | | Marketing (announcement, brochure,  
|      | | | slide-show presentation),  
|      | |  
|      | | Release (introduction,  
|      | | | release notes,  
|      | | | software user's manual,  
|      | | | terms & conditions,  
|      | | | dictionary),  
|      | |  
|      | | Third-party Resources  
|      | |  
|      | +-- "README5-EngineeringNotebook.txt"
```



```

|      +--- ["ProjectNotebook"] (Original Site-Package)
|      |
|      |      Contains a Toolkit User oriented collection of
|      |      ["EngineeringNotebook"] abstracts:
|      |
|      |      Project (purpose,
|      |                  goals,
|      |                  non-goals,
|      |                  features,
|      |                  capabilities,
|      |                  limitations)
|      |
|      |      +--- "README5-ProjectNotebook.txt"
|      |
|      +--- "README5-Notebooks.txt"
|
+--- ["SourceDistributions"] (Original)
|
|      Contains a collection of computer program
|      source code files that the Toolkit recip-
|      ient will need to install, operate, modify
|      and re-distribute the Toolkit.
|
+--- ["Developer-Sandboxes"] (Pre-dates Site-Packages)
|
|      A sandbox is a testing environment that iso-
|      lates untested code changes and outright
|      experimentation from the production environ-
|      ment or repository.
|
+--- ["tsWxGTUI_PyVx"] (Developer-Sandbox)
|
|      +--- ["Documents"] (Copy)
|      |
|      +--- ["ManPages"] (Copy)
|      |
|      +--- ["Python-2x"] (Developer-Sandbox)
|      |
|      |      +--- ["tsWxGTUI_Py2x"]
|      |
|      +--- ["Python-3x"] (Developer-Sandbox,
|      |                  Ported from Python-2x)
|      |
|      |      +--- ["tsWxGTUI_Py3x"]
|      |
|      +--- "README7-DeveloperSandboxes.txt"
|
+--- ["Site-Packages"]
|
|      Site-packages is the location where third-
|      party packages are installed (i.e., those
|      not part of the core Python distribution).
|      NOTE: That with Linux, Mac OS X and Unix
|      operating systems one must have root priv-
|      ileages to write to that location.

```

```
| | | +-- ["tsWxGTUI_PyVx"] (Site-Package)
| | | |
| | | +-- ["Documents"] (Copy)
| | | |
| | | +-- ["ManPages"] (Copy)
| | | |
| | | +-- ["Python-2x"] (Site-Package)
| | | |
| | | | +-- ["tsWxGTUI_Py2x"]
| | | |
| | | +-- ["Python-3x"] (Site-Package,
| | | | Ported from Python-2x)
| | | |
| | | +-- ["tsWxGTUI_Py3x"]
| | |
| | +-- "README8-SitePackages.txt"
|
+-- "README6-SourceDistributions.txt"
|
+-- "MANIFEST.in"
|
| Deliverable File inclusion criteria list.
|
+-- "MANIFEST_template.in"
|
| Deliverable Generic file inclusion criteria list
| template for any Python version-specific TeamSTARS
| "tsWxGTUI_PyVx" Toolkit.
|
+-- "MANIFEST_TREE.html"
|
| Non-Deliverable Diagram (Multi-Level Org Chart)
| depicting the hierarchical relationship between files
| in the release, in Hypertext Markup Language format.
|
| Diagram created via Command "./MANIFEST_TREE.sh".
|
+-- "MANIFEST_TREE.sh"
|
| Deliverable POSIX-style Command Line Interface shell
| script to generate diagrams depicting the hierarchical
| relationship between files in the release
| ("MANIFEST_TREE.html" and "MANIFEST_TREE.txt").
|
+-- "MANIFEST_TREE.txt"
|
| Non-Deliverable Diagram (Multi-Level Org Chart)
| depicting the hierarchical relationship between
| files in the release, in Plain Text format.
|
| Diagram created via Command "./MANIFEST_TREE.sh".
|
+-- "extract_tsWxGTUI_PyVx_Repository_zip_file.sh"
|
| Deliverable POSIX-style Command Line Interface shell
| script to extract contents of downloadable "zip" file
```

```

|
+-- "runLynx_Dump_Announcement.sh"
|
|   Deliverable POSIX-style Command Line Interface shell
|   script to convert a Web Page HTML version of the release
|   announcement into the same plain text format as other
|   files in the Documents sub-directory.
|
+-- "setup_tsWxGTUI_PyVx_Repository_tar_file.sh"
|
|   Deliverable POSIX-style Command Line Interface shell
|   script to generate downloadable "tarball" file.
|
+-- "setup_tsWxGTUI_PyVx_Repository_zip_file.sh"
|
|   Deliverable POSIX-style Command Line Interface shell
|   script to generate downloadable "zip" file.
|
+-- "README.txt"

```

===== Repository =====

1. Repository

Excerpt From Wikipedia, the free encyclopedia:

"A software repository is a storage location from which software packages may be retrieved and installed on a computer."

This is the repository for the TeamSTARS "tsWxGTUI_PyVx" Toolkit. It is the collection of documentation and computer program source code files that is being distributed by its author in order to publish and share the intellectual property with others.

It contains the following subdirectories and files:

1.1 Documents

Contains introductory and other training information for installers, developers, operators, troubleshooters and distributors.

Toolkit software documentation is written in plain text that accompanies computer software. It either explains how it operates or how to use it, and may mean different things to people in different roles. Types of software documentation include:

- a) Requirements - Statements that identify attributes, capabilities, characteristics, or qualities of a system. This is the foundation for what shall be or has been implemented.
- b) Architecture/Design - Overview of software.

Includes relations to an environment and construction principles to be used in design of software components.

- c) Technical - Documentation of code, algorithms, interfaces, and APIs.
- d) End user - Manuals for the end-user, system administrators and support staff.

1.2 ManPages

Contains a collection of man pages. A man page (short for manual page) is a form of online software documentation usually found on a Unix or Unix-like operating system. Topics covered include computer programs (including library and system calls), formal standards and conventions, and even abstract concepts.

Categories of man pages include:

- a) tsManPagesLibCLI
- b) tsManPagesLibGUI
- c) tsManPagesTestsLibCLI
- d) tsManPagesTestsLibGUI
- e) tsManPagesTestsToolsLibCLI
- f) tsManPagesTestsToolsLibGUI (Future)
- g) tsManPagesToolsCLI
- h) tsManPagesToolsGUI (Future)
- i) tsManPagesUtilities (Future)

1.3 Notebooks

Contains a collection of commentaries that express opinions or offerings of explanations about events or situations that might be useful to Toolkit installers, developers, operators, troubleshooters and distributors. The documents may be in Application-specific formats (Adobe PDF, JPEG Bit-mapped image, Microsoft Office, Plain text etc.).

The collection includes:

- a) DeveloperDocuments (API-References-Pixel-Mode-wxPython and Developer-ReadMe-Files); and
- b) EngineeringDocuments (Product Marketing Documentation, Project Documentation and Technical Documentation).

1.4 SourceDistributions

Contains a collection of computer program source code files that the Toolkit recipient will need to re-create the Toolkit.

The collection includes Developer-Sandbox and Site-Package versions of the following:

1.4.1 tsWxGTUI_Py2x

The source code files appropriate for use with Python 2.4-2.7.

1.4.2 tsWxGTUI_Py3x

The source code files appropriate for use with Python 3.0-3.4. These files are generated by converting their Python 2x counterparts with the 2to3 translation utility followed by debugging of unresolved syntax and type conversion issues.

----- NOTES:

- a) With Linux, Mac OS X and Unix operating systems one must have "root" or administrator privileges to write to the site-package location.
- b) If the user will not have permission to directly access the Repository but has a need to know specific contents, the system administrator should copy the appropriate contents of the Documents, ManPages and Notenook directories into each Site-Package.

The copies were not included in the distribution in order to:

- (1) avoid increasing the release development and qualification efforts; and
- (2) minimize the size of the downloadable "tar" and "zip" files.
- c) Users of third-party site-packages must explicitly import via its path from top-level package through lower-level packages to module:

site-package.package.module

- d) Examples for Python 2.4-2.7 site-packages:

```
from tsWxGTUI_Py2x.tsLibCLI import tsCxGlobals
from tsWxGTUI_Py2x.tsLibCLI import tsPlatformRunTimeEnvironment
from tsWxGTUI_Py2x.tsLibCLI import tsExceptions as tse
from tsWxGTUI_Py2x.tsLibCLI import tsLogger

from tsWxGTUI_Py2x.tsLibGUI import tsWx as wx
```

d) Examples for Python 3.0-3.4 site-packages:

```
from tsWxGTUI_Py3x.tsLibCLI import tsCxGlobals
from tsWxGTUI_Py3x.tsLibCLI import tsPlatformRunTimeEnvironment
from tsWxGTUI_Py3x.tsLibCLI import tsExceptions as tse
from tsWxGTUI_Py3x.tsLibCLI import tsLogger
```

```
from tsWxGTUI_Py3x.tsLibGUI import tsWx as wx
-----
```

1.5 Manifest

For a listing of the repository contents (complete with last modified date, time, size and access permissions), see:

```
"./MANIFEST_TREE.html"
"./MANIFEST_TREE.txt"
```

For additional commentary on the repository contents see:

```
"./Documents/README-Manifest.txt"
```

```
===== End-Of-File =====
```

Draft

4.1.1 Documents

```
#-----
#"Time-stamp: <09/30/2015 10:14:01 AM rsg>"
#-----

===== File: README3-Documents.txt =====

+-----+-----+ TeamSTARS "tsWxGTUI_PyVx" Toolkit
| ts | Wx |      with Python 2x & Python 3x based
+-----+-----+      Command Line Interface (CLI)
| G T U I |      and "Curses"-based "wxPython"-style,
+-----+-----+      Graphical-Text User Interface (GUI)

Get that cross-platform, pixel-mode "wxPython" feeling
on character-mode 8-/16-color (xterm-family) & non-color
(vt100-family) terminals and terminal emulators.

You can find this and other plain-text files in the
Toolkit subdirectory named:

    "./<Toolkit Recipient's Repository>/Documents".

<Your Working Repository>
(e.g. "tsWxGTUI_PyVx_Repository")
|
|
+-- ["Documents"]
|
|   This directory contains a collection of files
|   which provide the Toolkit recipient with an
}   understanding of the purpose, goals & capabil-
|   ities, non-goals & limitations, terms & condi-
|   tions and procedures for installing, operating,
|   modifying and redistributing the Toolkit.
|
+-- ["Announcement"]
|
|   Since embedded systems typically do not
|   have access to the engineering Notebooks,
|   this directory contains a collection of
|   files converted from:
|
|       tsWxGTUI_Vol.__0_SDIST_Announcement.doc.
|
|   The following files are usable on character-
|   mode terminals having only a monospaced font
|   of a single type and size (each conversion
|   introduced its own formatting compromises).
|
|   Lynx, the character-mode WEB browser and Hypertext
|   Markup Language file viewer, produced the only
|   conversion which retained information (but not the
|   layout) originally appearing in tables.
```

```
|
|
|      +-- "Announcement.txt" (Lynx HTM Dump with Layout File)
|
|      +-- "Announcement.ans" (ANSI Text with Layout File)
|      +-- "Announcement.asc" (MSDOS Text with Layout File)
|
|      |
|      |   The following files are usable on graphic-mode
|      |   terminals having proportional spaced fonts of
|      |   various types and sizes.
|      |
|      |   The graphic-mode application program, associated
|      |   with the file extension, retain the original
|      |   information and layout.
|      |
|      +-- "Announcement.pdf" (Adobe Portable Document File)
|      +-- "Announcement.htm" (Hypertext Markup Language File)
|      +-- "Announcement.rtf" (Rich Text Format File)
|
|      +-- "Announcement.doc" (Microsoft Word Document File)
|      +-- "4226.jpg"         (JPEG Masthead Image File)
|
| +-- "AUTHORS.txt"
| +-- "BUGS.txt"
| +-- "CHANGE_LOG.txt"
| +-- "CONFIGURE.txt"
| +-- "COPYING.txt"
| +-- "COPYRIGHT.txt"
| +-- "CREDITS.txt"
| +-- "DEMO.txt"
| +-- "FAQ.txt"
| +-- "GETTING_STARTED.txt"
| +-- "INSTALL.txt"
| +-- "LICENSE.txt"
| +-- "NEWS.txt"
| +-- "NOTICES.txt"
| +-- "OPERATE.txt"
| +-- "README.txt"
| +-- "README1-Introduction.txt"
| +-- "README2-Repository.txt"
| +-- "README3-Documents.txt"
| +-- "README4-ManPages.txt"
| +-- "README5-Notebooks.txt"
| +-- "README6-SourceDistributions.txt"
| +-- "README7-DeveloperSandboxes.txt"
| +-- "README8-SitePackages.txt"
| +-- "README9-KeyBoardMouseInput.txt"
| +-- "THANKS.txt"
| +-- "TO-DO.txt"
| +-- "TROUBLESHOOT.txt"
|
| +-- ["ManPages"]
|
| +-- ["Notebooks"]
|
| +-- ["SourceDistributions"]
|
```



```

+-- "README.txt"

===== ["Documents"] =====

This directory contains a collection of files which provide the Toolkit recipient with an understanding of the purpose, goals & capabilities, non-goals & limitation, terms & conditions and procedures and for installing, operating, modifying and redistributing the Toolkit.

"Announcement.htm" --- Introduces prospective and new
"Announcement.pdf"   recipients to the purpose, goals,
"Announcement.rtf"   non-goals, design and features of
"Announcement.txt"   this computer software. Provided
                     in single and multi-font formats.

"AUTHORS.txt"        --- List of the principal "tsWxGTUI_PyVx"
                     Toolkit author(s) and authors
                     credited for work covered by a prior
                     copyright and license.

"BUGS.txt"          --- List of Known Problems / Issues.

"CHANGE_LOG.txt"    --- List of Additions, Modification and
                     Deletions.

"CONFIGURE.txt"     --- Instructions for applying factory and
                     site-specific configurations.

"COPYING.txt"       --- Instructions for copying all or a
                     portion of the distribution.

"COPYRIGHT.txt"     --- Declaration of the exclusive legal
                     right, given to an originator or an
                     assignee to use, copy and distribute
                     computer software, and to authorize
                     others to do the same.

"CREDITS.txt"       --- Acknowledgment given to those whose
                     Copyrighted Work is used in accordance
                     with it's originator's Copyright and
                     License.

"DEMO.txt"          --- Narrated script demonstrating how to
                     install, configure, operate and trouble-
                     shoot the TeamSTARS "tsWxGTUI_PyVx"
                     Toolkit.

"FAQ.txt"           --- Answers to Frequently Asked
                     Questions.

"GETTING_STARTED.txt" --- Introduces new recipients to
                     the system requirements and
                     third-party resources avail-
                     able to new Toolkit users.

```

"INSTALL.txt" --- Describes steps to download, extract install and configure the "tsWxGUI" Toolkit.

"LICENSE.txt" --- General and special arrangements, provisions, rules, specifications and standards that form an integral part of the agreement or contract between the creator and recipient of Copyrighted and Licensed Work.

"MANIFEST.txt" --- Tally List for deliverable items.

"NEWS.txt" --- Announcements of new releases.

"NOTICES.txt" --- Details the copyright(s) and license(s).

"OPERATE.txt" --- Describes steps to use the "tsWxGUI" Toolkit.

"README.txt" --- Introduces new recipients to the purpose, goals, non-goals, design and features of the computer software product. Supplements include the following:

"README1-Introduction.txt"
"README2-Repository.txt"
"README3-Documents.txt"
"README4-ManPages.txt"
"README5-Notebooks.txt"
"README6-SourceDistributions.txt"
"README7-DeveloperSandboxes.txt"
"README8-SitePackages.txt"
"README9-KeyBoardMouseInput.txt"

"THANKS.txt" --- Acknowledgments to those otherwise unsung heroes who contributed time and effort to supporting the authors as planners, editors, designers, coders and testers.

"TO-DO.txt" --- A To-Do-List provides a roadmap for development and troubleshooting work.

"TROUBLESHOOT.txt" --- Provides a list of available reference resources and a guide for planning, developing and troubleshooting a cross-platform system of hundreds of files each containing a few, tens or hundred of class, data and method definitions. Its complexity becomes apparent in the recent software Lines-Of-Code metrics.

===== End-Of-File =====

Draft

4.1.2 ManPages

```
#-----
#"Time-stamp: <08/26/2015  4:33:57 PM rsg>"
#-----

===== File: README4-ManPages.txt =====

+-----+-----+ TeamSTARS "tsWxGTUI_PyVx" Toolkit
| ts | Wx |      with Python 2x & Python 3x based
+-----+-----+      Command Line Interface (CLI)
| G T U I |      and "Curses"-based "wxPython"-style,
+-----+-----+      Graphical-Text User Interface (GUI)

Get that cross-platform, pixel-mode "wxPython" feeling
on character-mode 8-/16-color (xterm-family) & non-color
(vt100-family) terminals and terminal emulators.

You can find this and other plain-text files in the
Toolkit subdirectory named:

    "<Toolkit Recipient's Repository>/Documents".

<Your Working Repository>
(e.g. "tsWxGTUI_PyVx_Repository")
|
+-- ["Documents"]
|
+-- ["ManPages"]
|
|   Deliverable Toolkit manual pages are a
|   form of online software documentation
|   usually found on a Unix or Unix-like oper-
|   ating system.
|
|   Topics covered include computer programs
|   (including library and system calls),
|   formal standards and conventions, and even
|   abstract concepts.
|
|   A user may NOT invoke a man page by issu-
|   ing the man command. Instead, a user may
|   display a man page by issuing the
|   less <man document file> command.
|
+-- ["tsManPagesLibCLI"]
|
|   +-- "runPydoc_tsManPagesCLI.sh"
|   +-- "runPyLint_tsManPagesCLI.sh"
|   +-- "tsApplication.man"
|   +-- "tsCommandLineEnv.man"
|   +-- "tsCommandLineInterface.man"
|   +-- "tsCxGlobals.man"
|   +-- "tsDoubleLinkedList.man"
```

```

+-- "tsExceptions.man"
+-- "tsGistGetTerminalSize.man"
+-- "tsLogger.man"
+-- "tsOperatorSettingsParser.man"
+-- "tsPlatformRunTimeEnvironment.man"
+-- "tsReportUtilities.man"
+-- "tsSysCommands.man"

+-- ["tsManPagesLibGUI"]
|
+-- "runPydoc_tsManPagesGUI.sh"
+-- "runPylint_tsManPagesGUI.sh"
+-- "tsWx.man"
+-- "tsWxAcceleratorEntry.man"
+-- "tsWxAcceleratorTable.man"
+-- "tsWxApp.man"
+-- "tsWxBoxSizer.man"
+-- "tsWxButton.man"
+-- "tsWxCallLater.man"
+-- "tsWxCaret.man"
+-- "tsWxCheckBox.man"
+-- "tsWxChoice.man"
+-- "tsWxColor.man"
+-- "tsWxColorDatabase.man"
+-- "tsWxControl.man"
+-- "tsWxControlWithItems.man"
+-- "tsWxCursor.man"
+-- "tsWxDebugHandlers.man"
+-- "tsWxDialog.man"
+-- "tsWxDialogButton.man"
+-- "tsWxDisplay.man"
+-- "tsWxDoubleLinkedList.man"
+-- "tsWxEraseEvent.man"
+-- "tsWxEvent.man"
+-- "tsWxEventDaemon.man"
+-- "tsWxEventLoop.man"
+-- "tsWxEventLoopActivator.man"
+-- "tsWxEventQueueEntry.man"
+-- "tsWxEventTableEntry.man"
+-- "tsWxEvtHandler.man"
+-- "tsWxFlexGridSizer.man"
+-- "tsWxFocusEvent.man"
+-- "tsWxFrame.man"
+-- "tsWxFrameButton.man"
+-- "tsWxGauge.man"
+-- "tsWxGlobals.man"
+-- "tsWxGraphicalTextUserInterface.man"
+-- "tsWxGridBagSizer.man"
+-- "tsWxGridSizer.man"
+-- "tsWxItemContainer.man"
+-- "tsWxKeyboardState.man"
+-- "tsWxKeyEvent.man"
+-- "tsWxListBox.man"
+-- "tsWxLog.man"
+-- "tsWxMenu.man"
+-- "tsWxMenuBar.man"

```

```
|      +-- "tsWxMouseEvent.man"  
|      +-- "tsWxMouseState.man"  
|      +-- "tsWxMultiFrameEnv.man"  
|      +-- "tsWxNonLinkedList.man"  
|      +-- "tsWXObject.man"  
|      +-- "tsWxPanel.man"  
|      +-- "tsWxPasswordEntryDialog.man"  
|      +-- "tsWxPoint.man"  
|      +-- "tsWxPyApp.man"  
|      +-- "tsWxPyEventBinder.man"  
|      +-- "tsWxPyOnDemandOutputWindow.man"  
|      +-- "tsWxPySimpleApp.man"  
|      +-- "tsWxPySizer.man"  
|      +-- "tsWxRadioBox.man"  
|      +-- "tsWxRadioButton.man"  
|      +-- "tsWxRect.man"  
|      +-- "tsWxScreen.man"  
|      +-- "tsWxScrollBar.man"  
|      +-- "tsWxScrollBarButton.man"  
|      +-- "tsWxScrollBarGauge.man"  
|      +-- "tsWxScrolled.man"  
|      +-- "tsWxScrolledText.man"  
|      +-- "tsWxScrolledWindow.man"  
|      +-- "tsWxShowEvent.man"  
|      +-- "tsWxSize.man"  
|      +-- "tsWxSizer.man"  
|      +-- "tsWxSizerItem.man"  
|      +-- "tsWxSizerItemList.man"  
|      +-- "tsWxSizerSpacer.man"  
|      +-- "tsWxSlider.man"  
|      +-- "tsWxSplashScreen.man"  
|      +-- "tsWxStaticBox.man"  
|      +-- "tsWxStaticBoxSizer.man"  
|      +-- "tsWxStaticLine.man"  
|      +-- "tsWxStaticText.man"  
|      +-- "tsWxStatusBar.man"  
|      +-- "tsWxSystemSettings.man"  
|      +-- "tsWxTaskBar.man"  
|      +-- "tsWxTextCtrl.man"  
|      +-- "tsWxTextEditBox.man"  
|      +-- "tsWxTextEntryDialog.man"  
|      +-- "tsWxTimer.man"  
|      +-- "tsWxToggleButton.man"  
|      +-- "tsWxTopLevelWindow.man"  
|      +-- "tsWxValidator.man"  
|      +-- "tsWxWindow.man"  
  
+-- ["tsManPagesTestsLibCLI"]  
|   |  
|   +-- "buildManPagesTestsCLI.sh"  
|   +-- "test_tsApplication.man"  
|   +-- "test_tsCommandLineEnv.man"  
|   +-- "test_tsDoubleLinkedList.man"  
|   +-- "test_tsOperatorSettingsParser.man"  
|   +-- "test_tsPlatformRunTimeEnvironment.man"  
|   +-- "test_tsSysCommand.man"
```

```

|
|   +-- ["tsManPagesTestsLibGUI"]
|   |   |
|   |   +-- "buildManPagesTestsGUI.sh"
|   |   +-- "test_tsWxBoxSizer.man"
|   |   +-- "test_tsWxCheckBox.man"
|   |   +-- "test_tsWxDisplay.man"
|   |   +-- "test_tsWxDoubleLinkedList.man"
|   |   +-- "test_tsWxGlobals.man"
|   |   +-- "test_tsWxGraphicalTextUserInterface.man"
|   |   +-- "test_tsWxGridSizer.man"
|   |   +-- "test_tsWxMultiFrameEnv.man"
|   |   +-- "test_tsWxRSM.man"
|   |   +-- "test_tsWxScrolledWindow.man"
|   |   +-- "test_tsWxScrolledWindowDual.man"
|   |   +-- "test_tsWxSplashScreen.man"
|   |   +-- "test_tsWxWidgets.man"
|   |
|   +-- ["tsManPagesToolsCLI"]
|   |   |
|   |   +-- "buildManPagesToolsCLI.sh"
|   |   +-- "runPydoc_tsManPagesToolsCLI.sh"
|   |   +-- "runPylint_tsManPagesToolsCLI.sh"
|   |   +-- "tsLinesOfCodeProjectMetrics.man"
|   |   +-- "tsPlatformQuery.man"
|   |   +-- "tsStripComments.man"
|   |   +-- "tsStripLineNumbers.man"
|   |   +-- "tsTreeCopy.man"
|   |   +-- "tsTreeTrimLines.man"
|   |
|   +-- ["tsManPagesToolsGUI"] (Future)
|   |   |
|   |   +-- To-Be-Determined
|   |
|   +-- ["tsManPagesToolsLibCLI"]
|   |   |
|   |   +-- To-Be-Determined
|   |
|   +-- ["tsManPagesToolsLibGUI"] (Future)
|   |   |
|   |   +-- To-Be-Determined
|   |
|   +-- ["tsManPagesUtilitiesCLI"] (Future)
|   |   |
|   |   +-- To-Be-Determined
|   |
|   +-- "README4-ManPages.txt"
|
+-- ["Notebooks"]
|
+-- ["SourceDistributions"]
|
+-- "README.txt"

```

===== TABLE OF CONTENTS =====

1. ["ManPages"]
2. How to create and install a manpage (Future)

===== ["ManPages"] =====

1. ["ManPages"]

The following defines the purpose and use of a set of on-line reference documents. This Toolkit provides utility scripts that:

- a) create rudimentary ManPages from source code;
- b) do NOT yet merge Toolkit ManPages with ManPages installed by the host computer operating system or with the installation of other third-pary add-ons.

Excerpt From Wikipedia, the free encyclopedia:

"A man page (short for manual page) is a form of online software documentation usually found on a Unix or Unix-like operating system. Topics covered include computer programs (including library and system calls), formal standards and conventions, and even abstract concepts. A user may invoke a man page by issuing the man command.

By default, man typically uses a terminal pager program such as more or less to display its output.

Usage

To read a manual page for a Unix command, type:

```
man <command_name>
```

Pages are traditionally referred to using the notation "name(section)": for example, ftp(1). The same page name may appear in more than one section of the manual, such as when the names of system calls, user commands, or macro packages coincide. Examples are man(1) and man(7), or exit(2) and exit(3).

The syntax for accessing the non-default manual section varies between different man implementations. On Solaris, for example, the syntax for reading printf(3C) is:

```
man -s 3c printf
```

On Linux and BSD derivatives the same invocation would be:

```
man 3 printf
```

which searches for printf in section 3 of the man pages."

===== "How to create and install a manpage" =====

2. How to create and install a manpage (Future)

The following may be useful for a future Toolkit enhancement.

Excerpts from Google Search:

"HowTo: Linux / UNIX Create a Manpage - nixCraft
www.cyberciti.biz/faq/linux-unix-creating-a-manpage/
 May 6, 2010 - How do I create a man page for my shell
 or python script under Linux / UNIX ...
 install -g 0 -o 0 -m 0644 nuseradd.1 /usr/local/man/man8/ gzip ...

How to create a manpage? - Ask Ubuntu
askubuntu.com/questions/42923/how-to-create-a-manpage
 Ask Ubuntu
 May 15, 2011 - With the help of Gmanedit · Install
 gmanedit you are able to create manpages with a
 graphical GUI. Gtk+ Manpages Editor is an editor for man ...

Linux Man Page Howto - Who is Jens Schweikhardt?
www.schweikhardt.net/man_page_howto.html
 The next decision is the directory in which it will
 finally be installed (say, when the user runs `make
 install` for your package.) On Linux, all man pages
 are below ...

Linux Howtos: System -> Creating Your Own MAN Page
www.linuxhowtos.org/system/creatingman.htm
 We will be using groff macros to create our manual page.
 These macros always ... Normally you put the version
 number of your program here. [center header]

How can I add man page entries for my own power tools?
unix.stackexchange.com/.../how-can-i-add-man-page-ent...
 Stack Exchange
 Feb 4, 2011 - I have no idea about how I can make my
 home-grown specialist scripts ... and you can create
 a man page from the POD file with the pod2man ... You
 can then optionally (b|g)zip it and put it in the
 appropriate man directory.

How to add entry in Linux man page database - Stack ...
stackoverflow.com/.../how-to-add-entry-in-linux-man-page-database
 Dec 25, 2012 - I have a manual page for mongoose web
 server named as mongoose.1 as a result of doing make
 and make install command to install ...

How should a formatted man page look?
www.tldp.org/HOWTO/Man-Page/q3.html
 Linux Documentation Project
 Here comes the man page for the (hypothetical)
 foo program. ... However, if you install using

'make prefix=/opt/gnu' the references in the man
page change to ..."

===== End-Of-File =====

Draft

4.1.3 Notebooks

```
#-----
#"Time-stamp: <08/26/2015  4:32:16 PM rsg>"
#-----

===== File: README5-Notebooks.txt =====

+-----+-----+ TeamSTARS "tsWxGTUI_PyVx" Toolkit
| ts | Wx |      with Python 2x & Python 3x based
+-----+-----+      Command Line Interface (CLI)
| G T U I |      and "Curses"-based "wxPython"-style,
+-----+-----+      Graphical-Text User Interface (GUI)

Get that cross-platform, pixel-mode "wxPython" feeling
on character-mode 8-/16-color (xterm-family) & non-color
(vt100-family) terminals and terminal emulators.

You can find this and other plain-text files in the
Toolkit subdirectory named:

    "./<Toolkit Recipient's Repository>/Documents".

<Your Working Repository>
(e.g. "tsWxGTUI_PyVx_Repository")
|
+-- ["Documents"]
|
+-- ["ManPages"]
|
+-- ["Notebooks"]
|
|   Contains a collection of commentaries that
|   express opinions or offerings of explana-
|   tions about events or situations that might
|   be useful to Toolkit installers, developers,
|   operators, troubleshooters and distributors.
|   The documents may be in Application-specific
|   formats (Adobe PDF, JPEG Bit-mapped image,
|   Microsoft Office, Plain text etc.).
|
|   The collection includes:
|
+-- ["DeveloperNotebook"]
|
|   +-- ["DeveloperReadMeFiles"]
|
|       Repository for entire collection of
|       README and topic of interest files
|
+-- ["API"]
|
|   Application Programming Interfaces
|
|
```

```
| | | +-- ["API-tsLibCLI"]  
| | | |  
| | | | Library of Character-mode  
| | | | Command Line Interface Building Blocks  
| | | |  
| | | +-- ["API-tsLibGUI"]  
| | | |  
| | | | Library of Character-mode  
| | | | Graphical User Interface Building Blocks  
| | | |  
| | | +-- ["API-wxPython-2.8.9.2-docs"]  
| | | |  
| | | | Library of Pixel-Mode  
| | | | wxPython-2.8.9.2 Building Blocks  
| | | | To-Be-Emulated  
| | | |  
| | | +-- ["API-wxPython-3.0.2.0-docs"]  
| | | |  
| | | | Future Library of Pixel-Mode  
| | | | wxPython-3.0.2.0 Building Blocks  
| | | | To-Be-Emulated  
| | | |  
+-- ["CLI-How-To-Files"]  
| | |  
| | | Tutorial on Python programing language  
| | | generation evolution, differences and  
| | | configuration control  
| | |  
| | | +-- "CLI_0_hello_world_print_statement.py"  
| | | +-- "CLI_1_hello_world_print_function.py"  
| | | +-- "CLI_2_hello_world_script_environment.py"  
| | | +-- "CLI_3_hello_world_main_module_application.py"  
| | | +-- "test_tsCxGlobals.py"  
| | |  
+-- ["GUI-How-To-Files"]  
| | |  
| | | Tutorial on Python graphical library  
| | | generation evolution, differences and  
| | | configuration control  
| | |  
| | | +-- "GUI_4_Curses_Widget-API-application.py"  
| | | +-- "GUI_5_tsWxGTUI_Widget-API-application.py"  
| | | +-- "GUI_6_tsWxGTUI_BoxSizer-API-application.py"  
| | | +-- "test_tsWxGlobals.py"  
| | |  
+-- "README5-DeveloperNotebook.txt")  
  
+-- ["EngineeringNotebook"]  
| | |  
| | | Contains a collection of files, in application-  
| | | specific formats, which provide the Toolkit archi-  
| | | tect, product manager, project engineer, system  
| | | engineer, software engineer, test engineer and  
| | | troubleshooter with:  
| | |  
+-- ["Adobe-PDF-Files"]
```

```

|         |      +--- ["API-Preview"]
|         |      +--- ["ASCII-Text-Files"]
|         |      +--- ["Bugzilla-Products-and-Components"]
|         |      +--- ["JPEG-Image-Files"]
|         |      +--- ["MS-Access-Files"]
|         |      +--- ["MS-Excel-Files"]
|         |      +--- ["MS-PowerPoint-Files"]
|         |      +--- ["MS-Visio-Files"]
|         |      +--- ["MS-Word-Files"]
|         |      |
|         |      +--- "README-EngineeringNotebook.txt"
|         |
|         +--- ["ProjectNotebook"]
|         |
|         |      Contains a Toolkit User oriented collection of
|         |      ["EngineeringNotebook"] abstracts:
|         |
|         |      Project (purpose,
|         |                  goals,
|         |                  non-goals,
|         |                  features,
|         |                  capabilities,
|         |                  limitations)
|         |
|         |      +--- "README-ProjectNotebook.txt")
|         |
|         +--- "README5-Notebooks.txt"
|
+--- ["SourceDistributions"]
|
+--- "README.txt"

```

===== TABLE OF CONTENTS =====

1. ["Developer-Documents"]

- 1.1 ["API-Preview"]
- 1.2 ["API-References-Pixel-Mode-wxPython"] (Future)
- 1.3 ["DeveloperReadMeFiles"] (Future)

2. ["Engineering-Documents"]

- 2.1 Product Marketing Documentation (Future)
- 2.2 Project Documentation
- 2.3 Technical Documentation

===== ["Developer-Documents"] =====

1. ["Developer-Documents"]

This directory contains a collection of files which provide the Toolkit Building Block, Tool and Application programmer with an understanding of the design and usage of the Toolkit's CLI and GUI Application Programming In-

terface.

1.1 ["API-References-Pixel-Mode-wxPython"]

1.2 ["DeveloperReadMeFiles"]

===== ["Engineering-Documents"] =====

2. ["Engineering-Documents"]

This directory contains a collection of files, in application-specific formats, which provide the Toolkit architect, product manager, project engineer, system engineer, software engineer, test engineer and troubleshooter with:

2.1 Product Marketing Documentation

- a) announcement notice(s)
- b) brochure(s)
- c) introduction

2.2 Project Documentation

- a) goal and capability objectives
- b) non-goal and limitation constraints
- c) project plans

2.3 Technical Documentation

- a) architectural plans
- b) system specifications
- c) Interface specifications
- d) software specifications
- e) design specifications
- f) test specifications
- g) software user manual

===== End-Of-File =====

4.1.4 SourceDistributions

```
#-----
#"Time-stamp: <08/26/2015  4:35:18 PM rsg>"
#-----

===== File: README6-SourceDistributions.txt =====

+-----+-----+ TeamSTARS "tsWxGTUI_PyVx" Toolkit
| ts | Wx |      with Python 2x & Python 3x based
+-----+-----+      Command Line Interface (CLI)
| G T U I |      and "Curses"-based "wxPython"-style,
+-----+-----+      Graphical-Text User Interface (GUI)

Get that cross-platform, pixel-mode "wxPython" feeling
on character-mode 8-/16-color (xterm-family) & non-color
(vt100-family) terminals and terminal emulators.

You can find this and other plain-text files in the
Toolkit subdirectory named:

    "./<Toolkit Recipient's Repository>/Documents".

<Your Working Repository>
(e.g. "tsWxGTUI_PyVx_Repository")
|
+-- ["Documents"] (Original)
|
+-- ["ManPages"] (Original)
|
+-- ["Notebooks"] (Original)
|
+-- ["SourceDistributions"] (Original)
|
|   Contains a collection of computer program
|   source code files that the Toolkit recip-
|   ient will need to install, operate, modify
|   and re-distribute the Toolkit.
|
+-- ["Developer-Sandboxes"] (Pre-dates Site-Packages)
|
|   |   A sandbox is a testing environment that iso-
|   |   lates untested code changes and outright
|   |   experimentation from the production environ-
|   |   ment or repository.
|   |
|   +-- ["tsWxGTUI_PyVx"] (Developer-Sandbox)
|   |
|   |   +-- ["Documents"] (Copy)
|   |   |
|   |   +-- ["ManPages"] (Copy)
|   |   |
|   |   +-- ["Python-2x"] (Developer-Sandbox)
|   |   |
|   |   |
```

```
| | | +-- ["tsWxGTUI_Py2x"]  
| | | |  
| | | +-- ["Python-3x"] (Developer-Sandbox,  
| | | | Ported from Python-2x)  
| | | |  
| | | +-- ["tsWxGTUI_Py3x"]  
| | |  
+-- "README7-DeveloperSandboxes.txt"  
  
+-- ["Site-Packages"]  
| |  
| | Site-packages is the location where third-  
| | party packages are installed (i.e., those  
| | not part of the core Python distribution).  
| | NOTE: That with Linux, Mac OS X and Unix  
| | operating systems one must have root priv-  
| | ileages to write to that location.  
| |  
+-- ["tsWxGTUI_PyVx"] (Site-Package)  
| |  
| | +-- ["Documents"] (Copy)  
| | |  
| | +-- ["ManPages"] (Copy)  
| | |  
| | +-- ["Python-2x"] (Site-Package)  
| | |  
| | | +-- ["tsWxGTUI_Py2x"]  
| | |  
| | +-- ["Python-3x"] (Site-Package,  
| | | Ported from Python-2x)  
| | |  
| | +-- ["tsWxGTUI_Py3x"]  
| |  
+-- "README8-SitePackages.txt"  
  
+-- "README6-SourceDistributions.txt"
```

===== TABLE OF CONTENTS =====

1. Source Code
2. Developer-Sandbox
3. Site-Package

```
===== Source Code =====
```

- ## 1. Source Code

Excerpt From Wikipedia, the free encyclopedia:

"In computing, source code is any collection of computer instructions (possibly with comments) written using some human-readable computer language, usually as text. The source code of a program is specially designed to facilitate the work of computer programmers, who specify the

actions to be performed by a computer mostly by writing source code. The source code is often transformed by a compiler program into low-level machine code understood by the computer. The machine code might then be stored for execution at a later time. Alternatively, an interpreter can be used to analyze and perform the outcomes of the source code program directly on the fly.

Most computer applications are distributed in a form that includes executable files, but not their source code. If the source code were included, it would be useful to a user, programmer, or system administrator, who may wish to modify the program or to understand how it works.

Aside from its machine-readable forms, source code also appears in books and other media; often in the form of small code snippets, but occasionally complete code bases; a well-known case is the source code of PGP."

===== Developer-Sandbox =====

2. Developer-Sandbox

Excerpt From Wikipedia, the free encyclopedia:

"A sandbox is a testing environment that isolates untested code changes and outright experimentation from the production environment or repository, in the context of software development including Web development and revision control. Sandboxing protects "live" servers and their data, vetted source code distributions, and other collections of code, data and/or content, proprietary or public, from changes that could be damaging (regardless of the intent of the author of those changes) to a mission-critical system or which could simply be difficult to revert. Sandboxes replicate at least the minimal functionality needed to accurately test the programs or other code under development (e.g. usage of the same environment variables as, or access to an identical database to that used by, the stable prior implementation intended to be modified; there are many other possibilities, as the specific functionality needs vary widely with the nature of the code and the application[s] for which it is intended.)

The concept of the sandbox (sometimes also called a working directory, a test server or development server) is typically built into revision control software such as CVS and Subversion (SVN), in which developers "check out" a copy of the source code tree, or a branch thereof, to examine and work on. Only after the developer has (hopefully) fully tested the code changes in their own sandbox should the changes be checked back into and merged with the repository and thereby made available to other developers or end users of the software.[1]

By further analogy, the term "sandbox" can also be applied in computing and networking to other temporary or indefinite isolation areas, such as security sandboxes and search engine sandboxes (both of which have highly specific meanings), that prevent incoming data from affecting a "live" system (or aspects thereof) unless/until defined requirements or criteria have been met."

Unlike the contents of the installable site-package, this sandbox uses a multi-level tree of subdirectories and associated files whose topology is defined by a set of package "__init__.py" files which collaborate in performing dynamic path generation and importing of modules and subpackages. Applications import individual packages and individual modules simply by name (if module name is unique) or by package.module name (if module name is not unique).

===== Site-Package =====

3. Site-Package

Site-packages is the location where third-party packages are installed (i.e., those not part of the core Python distribution). NOTE: That with Linux, Mac OS X and Unix operating systems one must have root privileges to write to that location.

Unlike the contents of the Developer-Sandbox, the third-party site-package and its users must explicitly import via the site-package.package.module path identifier.

===== End-Of-File =====

4.1.4.1 Developer-Sandbox

```
#-----
#"Time-stamp: <08/26/2015  4:36:08 PM rsg>"
#-----
```

=== Title Page for File: README7-Developer-Sandboxes.txt ===

```
+-----+-----+ TeamSTARS "tsWxGTUI_PyVx" Toolkit
| ts | Wx |      with Python 2x & Python 3x based
+-----+-----+      Command Line Interface (CLI)
| G T U I |      and "Curses"-based "wxPython"-style,
+-----+-----+      Graphical-Text User Interface (GUI)
```

Get that cross-platform, pixel-mode "wxPython" feeling on character-mode 8-/16-color (xterm-family) & non-color (vt100-family) terminals and terminal emulators.

You can find this and other plain-text files in the Toolkit subdirectory named:

"/<Toolkit Recipient's Repository>/Documents".

```
<Your Working Repository>
(e.g. "tsWxGTUI_PyVx_Repository")
|
+-- ["Documents"]
|
+-- ["ManPages"]
|
+-- ["Notebooks"]
|
+-- ["SourceDistributions"]
|
|   Contains a collection of computer program
|   source code files that the Toolkit recip-
|   ient will need to install, operate, modify
|   and re-distribute the Toolkit.
|
+-- "README6-SourceDistributions.txt"
|
+-- ["Developer-Sandboxes"] (Pre-dates Site-Packages)
|
|   A sandbox is a testing environment that iso-
|   lates untested code changes and outright experi-
|   mentation from the production environment or
|   repository.
|
+-- ["tsWxGTUI_PyVx"]
|
|   Contains one or more Python language gener-
|   ation-specific releases each sharing the same
|   programmer (API) and user (CLI & GUI) inter-
|   faces, documents and manual pages.
|
```

```

+--- ["Documents"]
|
+--- ["ManPages"]
|
+--- ["Python-2x"]
|
|   Second generation Python programming
|   language.
|
+--- ["tsWxGTUI_Py2x"]
|
|   +--- ["tsDemoArchive"]
|   |
|   |   +--- ["src"]
|   |   |
|   |   +--- "TermsAndConditions.txt"
|   |
|   +--- ["tsLibCLI"]
|   |
|   |   +--- ["tsApplicationPkg"]
|   |   |
|   |   |   +--- ["src"]
|   |   |   +--- ["test"]
|   |   |
|   |   +--- ["tsCommandLineEnvPkg"]
|   |   |
|   |   |   +--- ["src"]
|   |   |   +--- ["test"]
|   |   |
|   |   +--- ["tsCommandLineInterfacePkg"]
|   |   |
|   |   |   +--- ["src"]
|   |   |   +--- ["test"]
|   |   |
|   |   +--- ["tsCxCGlobalsPkg"]
|   |   |
|   |   |   +--- ["src"]
|   |   |   +--- ["test"]
|   |   |
|   |   +--- ["tsDoubleLinkedListPkg"]
|   |   |
|   |   |   +--- ["src"]
|   |   |   +--- ["test"]
|   |   |
|   |   +--- ["tsExceptionPkg"]
|   |   |
|   |   |   +--- ["src"]
|   |   |   +--- ["test"]
|   |   |
|   |   +--- ["tsGistGetTerminalSizePkg"]
|   |   |
|   |   |   +--- ["src"]
|   |   |
|   |   +--- ["tsLoggerPkg"]
|   |   |
|   |   |   +--- ["src"]

```


[illegible]

```
|
|                                     +-- ["tsToolsGUI"]
|                                     |
|                                     +-- ["tsUtilities"]
|                                     |
| +-- ["Python-3x"] (Ported from Python-2x)
|     |
|     | Third generation Python programming
|     | language.
|     |
|     +-- ["tsWxGTUI_Py3x"]
|
+-- "README.txt"
```

===== End-Of-File =====

- 1** *Command Line Interface Library and Tools* (on page 109)
- 2** *Graphical Text User Interface Library and Tools* (on page 109)

4.1.4.1.1 Command Line Interface Library and Tools

The Command Line Interface Library (tsLibCLI) include the following computer software components (with availability as marked):

- 1 tsLibCLI/tsApplicationPkg/src/tsApplication.py (Available)
- 2 tsLibCLI/tsCommandLineEnvPkg/src/tsCommandLineEnv.py (Available)
- 3 tsLibCLI/tsCommandLineInterfacePkg/src/tsCommandLineInterface.py (Available)
- 4 tsLibCLI/tsCxGlobalsPkg/src/tsCxGlobals.py (Available)
- 5 tsLibCLI/tsDoubleLinkedListPkg/src/tsDoubleLinkedList.py (Available)
- 6 tsLibCLI/tsExceptionPkg/src/tsExceptions.py (Available)
- 7 tsLibCLI/tsLoggerPkg/src/tsLogger.py (Available)
- 8 tsLibCLI/tsOperatorSettingsParserPkg/src/tsOperatorSettingsParser.py (Available)
- 9 tsLibCLI/tsPlatformRunTimeEnvironmentPkg/src/tsPlatformRunTimeEnvironment.py (Available)
- 10 tsLibCLI/tsReportUtilityPkg/src/tsReportUtilities.py (Available)
- 11 tsLibCLI/tsSysCommandsPkg/src/tsSysCommands.py (Available)

4.1.4.1.2 Graphical Text User Interface Library and Tools

The Graphical Text User Interface Library (tsLibGUI) include the following computer software components (with availability as marked):

- 1 tsLibGUI/tsWxPkg/src/tsWx.py (Available)
- 2 tsLibGUI/tsWxPkg/src/tsWxAcceleratorEntry.py (Available)
- 3 tsLibGUI/tsWxPkg/src/tsWxAcceleratorTable.py (Available)
- 4 tsLibGUI/tsWxPkg/src/tsWxApp.py (Available)
- 5 tsLibGUI/tsWxPkg/src/tsWxBoxSizer.py (Available)
- 6 tsLibGUI/tsWxPkg/src/tsWxButton.py (Available)

- 7 tsLibGUI/tsWxPkg/src/tsWxCallLater.py (Future)
- 8 tsLibGUI/tsWxPkg/src/tsWxCaret.py (Available)
- 9 tsLibGUI/tsWxPkg/src/tsWxCheckBox.py (Available)
- 10 tsLibGUI/tsWxPkg/src/tsWxChoice.py (Future)
- 11 tsLibGUI/tsWxPkg/src/tsWxColor.py (Available)
- 12 tsLibGUI/tsWxPkg/src/tsWxColorDatabase.py (Available)
- 13 tsLibGUI/tsWxPkg/src/tsWxCommandLineEnv.py (Available)
- 14 tsLibGUI/tsWxPkg/src/tsWxControl.py (Available)
- 15 tsLibGUI/tsWxPkg/src/tsWxControlWithItems.py (Available)
- 16 tsLibGUI/tsWxPkg/src/tsWxCursor.py (Available)
- 17 tsLibGUI/tsWxPkg/src/tsWxDebugHandlers.py (Future)
- 18 tsLibGUI/tsWxPkg/src/tsWxDialog.py (Available)
- 19 tsLibGUI/tsWxPkg/src/tsWxDialogButton.py (Available)
- 20 tsLibGUI/tsWxPkg/src/tsWxDisplay.py (Available)
- 21 tsLibGUI/tsWxPkg/src/tsWxDoubleLinkedList.py (Available)
- 22 tsLibGUI/tsWxPkg/src/tsWxEraseEvent.py (Future)
- 23 tsLibGUI/tsWxPkg/src/tsWxEvent.py (Available)
- 24 tsLibGUI/tsWxPkg/src/tsWxEventDaemon.py (Future)
- 25 tsLibGUI/tsWxPkg/src/tsWxEventLoop.py (Available)
- 26 tsLibGUI/tsWxPkg/src/tsWxEventLoopActivator.py (Future)
- 27 tsLibGUI/tsWxPkg/src/tsWxEventQueueEntry.py (Future)
- 28 tsLibGUI/tsWxPkg/src/tsWxEventTableEntry.py (Available)
- 29 tsLibGUI/tsWxPkg/src/tsWxEvtHandler.py (Available)
- 30 tsLibGUI/tsWxPkg/src/tsWxFlexGridSizer.py (Future)
- 31 tsLibGUI/tsWxPkg/src/tsWxFocusEvent.py (Future)
- 32 tsLibGUI/tsWxPkg/src/tsWxFrame.py (Available)
- 33 tsLibGUI/tsWxPkg/src/tsWxFrameButton.py (Available)
- 34 tsLibGUI/tsWxPkg/src/tsWxGauge.py (Available)
- 35 tsLibGUI/tsWxPkg/src/tsWxGlobals.py (Available)
- 36 tsLibGUI/tsWxPkg/src/tsWxGraphicalTextUserInterface.py (Available)
- 37 tsLibGUI/tsWxPkg/src/tsWxGridBagSizer.py (Future)
- 38 tsLibGUI/tsWxPkg/src/tsWxGridSizer.py (Available)
- 39 tsLibGUI/tsWxPkg/src/tsWxItemContainer.py (Future)
- 40 tsLibGUI/tsWxPkg/src/tsWxKeyboardState.py (Available)

- 41 tsLibGUI/tsWxPkg/src/tsWxKeyEvent.py (Available)
- 42 tsLibGUI/tsWxPkg/src/tsWxLayoutAlgorithm.py (Future)
- 43 tsLibGUI/tsWxPkg/src/tsWxListBox.py (Future)
- 44 tsLibGUI/tsWxPkg/src/tsWxLog.py (Future)
- 45 tsLibGUI/tsWxPkg/src/tsWxMenu.py (Future)
- 46 tsLibGUI/tsWxPkg/src/tsWxMenuBar.py (Future)
- 47 tsLibGUI/tsWxPkg/src/tsWxMouseEvent.py (Available)
- 48 tsLibGUI/tsWxPkg/src/tsWxMouseState.py (Available)
- 49 tsLibGUI/tsWxPkg/src/tsWxMultiFrameEnv.py (Available)
- 50 tsLibGUI/tsWxPkg/src/tsWxObject.py (Available)
- 51 tsLibGUI/tsWxPkg/src/tsWxPanel.py (Available)
- 52 tsLibGUI/tsWxPkg/src/tsWxPoint.py (Available)
- 53 tsLibGUI/tsWxPkg/src/tsWxPyApp.py (Available)
- 54 tsLibGUI/tsWxPkg/src/tsWxPyEventBinder.py (Available)
- 55 tsLibGUI/tsWxPkg/src/tsWxPyOnDemandOutputWindow.py (Available)
- 56 tsLibGUI/tsWxPkg/src/tsWxPySimpleApp.py (Available)
- 57 tsLibGUI/tsWxPkg/src/tsWxPySizer.py (Available)
- 58 tsLibGUI/tsWxPkg/src/tsWxRadioBox.py (Available)
- 59 tsLibGUI/tsWxPkg/src/tsWxRadioButton.py (Available)
- 60 tsLibGUI/tsWxPkg/src/tsWxRect.py (Available)
- 61 tsLibGUI/tsWxPkg/src/tsWxScreen.py (Available)
- 62 tsLibGUI/tsWxPkg/src/tsWxScrollBar.py (Available)
- 63 tsLibGUI/tsWxPkg/src/tsWxScrollBarButton.py (Available)
- 64 tsLibGUI/tsWxPkg/src/tsWxScrollBarGauge.py (Available)
- 65 tsLibGUI/tsWxPkg/src/tsWxScrolled.py (Available)
- 66 tsLibGUI/tsWxPkg/src/tsWxScrolledText.py (Available)
- 67 tsLibGUI/tsWxPkg/src/tsWxScrolledWindow.py (Available)
- 68 tsLibGUI/tsWxPkg/src/tsWxShowEvent.py (Available)
- 69 tsLibGUI/tsWxPkg/src/tsWxSize.py (Available)
- 70 tsLibGUI/tsWxPkg/src/tsWxSizer.py (Available)
- 71 tsLibGUI/tsWxPkg/src/tsWxSizerFlags.py (Available)
- 72 tsLibGUI/tsWxPkg/src/tsWxSizerItem.py (Available)
- 73 tsLibGUI/tsWxPkg/src/tsWxSizerItemList.py (Available)
- 74 tsLibGUI/tsWxPkg/src/tsWxSizerSpacer.py (Available)

- 75** tsLibGUI/tsWxPkg/src/tsWxSlider.py (Future)
- 76** tsLibGUI/tsWxPkg/src/tsWxSplashScreen.py (Available)
- 77** tsLibGUI/tsWxPkg/src/tsWxStaticBox.py (Available)
- 78** tsLibGUI/tsWxPkg/src/tsWxStaticBoxSizer.py (Available)
- 79** tsLibGUI/tsWxPkg/src/tsWxStaticLine.py (Available)
- 80** tsLibGUI/tsWxPkg/src/tsWxStaticText.py (Available)
- 81** tsLibGUI/tsWxPkg/src/tsWxStatusBar.py (Available)
- 82** tsLibGUI/tsWxPkg/src/tsWxSystemSettings.py (Available)
- 83** tsLibGUI/tsWxPkg/src/tsWxTaskBar.py (Available)
- 84** tsLibGUI/tsWxPkg/src/tsWxTextCtrl.py (Available)
- 85** tsLibGUI/tsWxPkg/src/tsWxTimer.py (Future)
- 86** tsLibGUI/tsWxPkg/src/tsWxToggleButton.py (Future)
- 87** tsLibGUI/tsWxPkg/src/tsWxTopLevelWindow.py (Available)
- 88** tsLibGUI/tsWxPkg/src/tsWxValidator.py (Future)
- 89** tsLibGUI/tsWxPkg/src/tsWxWindow.py (Available)

4.1.4.2 Site-Package

```
#-----
#"Time-stamp: <08/26/2015  4:36:32 PM rsg>"
#-----

===== Title Page for File: README8-Site-Packages.txt =====

+-----+-----+ TeamSTARS "tsWxGTUI_PyVx" Toolkit
| ts | Wx |      with Python 2x & Python 3x based
+-----+-----+      Command Line Interface (CLI)
| G T U I |      and "Curses"-based "wxPython"-style,
+-----+-----+      Graphical-Text User Interface (GUI)

Get that cross-platform, pixel-mode "wxPython" feeling
on character-mode 8-/16-color (xterm-family) & non-color
(vt100-family) terminals and terminal emulators.

You can find this and other plain-text files in the
Toolkit subdirectory named:

    "./<Toolkit Recipient's Repository>/Documents".

<Your Working Repository>
(e.g. "tsWxGTUI_PyVx_Repository")
|
+-- ["Documents"]
|
+-- ["ManPages"]
|
+-- ["Notebooks"]
|
+-- ["SourceDistributions"]
|
|   Contains a collection of computer program
|   source code files that the Toolkit recip-
|   ient will need to install, operate, modify
|   and re-distribute the Toolkit.
|
|   +-- "README6-SourceDistributions.txt"
|
|   +-- ["Developer-Sandboxes"] (Pre-dates Site-Packages)
|
|       |
|       |   A sandbox is a testing environment that iso-
|       |   lates untested code changes and outright experi-
|       |   mentation from the production environment or
|       |   repository.
|       |
|       +-- ["tsWxGTUI_PyVx"]
|
|           |
|           |   Contains one or more Python language gener-
|           |   ation-specific releases each sharing the same
|           |   programmer (API) and user (CLI & GUI) inter-
|           |   faces, documents and manual pages.
|           |
|           |
```

```

+--- ["Documents"]
|
+--- ["ManPages"]
|
+--- ["Python-2x"]
|
|   Second generation Python programming
|   language.
|
+--- ["tsWxGTUI_Py2x"]
|
|   +--- ["tsDemoArchive"]
|   |
|   |   +--- ["src"]
|   |   |
|   |   +--- "TermsAndConditions.txt"
|   |
|   +--- ["tsLibCLI"]
|   |
|   |   +--- ["tsApplicationPkg"]
|   |   |
|   |   |   +--- ["src"]
|   |   |   +--- ["test"]
|   |   +--- ["tsCommandLineEnvPkg"]
|   |   |
|   |   |   +--- ["src"]
|   |   |   +--- ["test"]
|   |   +--- ["tsCommandLineInterfacePkg"]
|   |   |
|   |   |   +--- ["src"]
|   |   |   +--- ["test"]
|   |   +--- ["tsCxCGlobalsPkg"]
|   |   |
|   |   |   +--- ["src"]
|   |   |   +--- ["test"]
|   |   +--- ["tsDoubleLinkedListPkg"]
|   |   |
|   |   |   +--- ["src"]
|   |   |   +--- ["test"]
|   |   +--- ["tsExceptionPkg"]
|   |   |
|   |   |   +--- ["src"]
|   |   |   +--- ["test"]
|   |   +--- ["tsGistGetTerminalSizePkg"]
|   |   |
|   |   |   +--- ["src"]
|   |   +--- ["tsLoggerPkg"]
|   |   |
|   |   |   +--- ["src"]

```

		+-- ["test"]
		+--
["tsOperatorSettingsParserPkg"]		
		+-- ["src"]
		+-- ["test"]
		+--
["tsPlatformRunTimeEnvironmentPkg"]		
		+-- ["src"]
		+-- ["test"]
		+-- ["tsReportUtilityPkg"]
		+-- ["src"]
		+-- ["test"]
		+-- ["tsSysCommandsPkg"]
		+-- ["src"]
		+-- ["test"]
		+-- ["tsLibGUI"]
		+-- ["tsWxPkg"]
		+-- ["src"]
		+-- ["test"]
		+-- ["tsToolsCLI"]
		+--
["tsLinesOfCodeProjectMetricsPkg"]		
		+-- ["src"]
		+-- ["test"]
		+-- ["tsPlatformQueryPkg"]
		+-- ["src"]
		+-- ["test"]
		+-- ["tsStripCommentsPkg"]
		+-- ["src"]
		+-- ["test"]
		+-- ["tsStripLineNumbersPkg"]
		+-- ["src"]
		+-- ["test"]
		+-- ["tsTreeCopyPkg"]

```
|
|                                     +-- ["src"]
|                                     +-- ["test"]
|                                     |
|                                     +-- ["tsTreeTrimLinesPkg"]
|                                     |
|                                     +-- ["src"]
|                                     +-- ["test"]
|
|                                     +-- ["tsToolsGUI"]
|                                     |
|                                     +-- ["tsUtilities"]
|
+-- ["Python-3x"] (Ported from Python-2x)
|
|   Third generation Python programming
|   language.
|
+-- ["tsWxGTUI_Py3x"]
|
+-- ["Site-Packages"]
|
|   A site-packages is the location where third-
|   party packages are installed (i.e., those
|   not part of the core Python distribution).
|   NOTE: That with Linux, Mac OS X and Unix
|   operating systems one must have root priv-
|   ileages to write to that location.
|
+-- ["tsWxGTUI_PyVx"]
|
|   +-- ["Documents"]
|
|   +-- ["ManPages"]
|
|   +-- ["Python-2x"]
|       |
|       |   Second generation Python programming
|       |   language.
|       |
|       +-- ["tsWxGTUI_Py2x"]
|           |
|           +-- ["tsDemoArchive"]
|               |
|               +-- ["tsTestsLibCLI"]
|               +-- ["tsTestsLibGUI"]
|               +-- ["tsTestsToolsCLI"]
|               +-- ["tsTestsToolsGUI"]
|               +-- ["tsTestsToolsLibCLI"]
|               +-- ["tsTestsToolsLibGUI"]
|
|           +-- ["tsLibCLI"]
|
|           +-- ["tsLibGUI"]
|
|           +-- ["tsToolsCLI"]
```

```
|                                     +-- ["tsToolsGUI"]  
|                                     |  
|                                     +-- ["tsUtilities"]  
|                                     |  
+-- ["Python-3x"] (Ported from Python-2x)  
|                                     |  
|                                     | Third generation Python programming  
|                                     | language.  
|                                     |  
|                                     +-- ["tsWxGTUI_Py3x"]  
  
+-- "README.txt"
```

```
===== End-Of-File =====
```

- 1 *Command Line Interface Library and Tools* (on page 117)
- 2 *Graphical Text User Interface Library and Tools* (see "*Graphical Text User Interface Library and Tools (Draft)*" on page 117)

4.1.4.2.1 Command Line Interface Library and Tools

The Command Line Interface Library (tsLibCLI) include the following computer software components (with availability as marked):

- 1 tsLibCLI/tsApplication.py (Available)
- 2 tsLibCLI/tsCommandLineEnv.py (Available)
- 3 tsLibCLI/tsCommandLineInterface.py (Available)
- 4 tsLibCLI/tsCxGlobals.py (Available)
- 5 tsLibCLI/tsDoubleLinkedList.py (Available)
- 6 tsLibCLI/tsExceptionPkg/src/tsExceptions.py (Available)
- 7 tsLibCLI/tsLoggerPkg/src/tsLogger.py (Available)
- 8 tsLibCLI/tsOperatorSettingsParser.py (Available)
- 9 tsLibCLI/tsPlatformRunTimeEnvironment.py (Available)
- 10 tsLibCLI/tsReportUtilities.py (Available)
- 11 tsLibCLI/tsSysCommands.py (Available)

4.1.4.2.2 Graphical Text User Interface Library and Tools (Draft)

The Graphical Text User Interface Library (tsLibGUI) include the following computer software components (with availability as marked):

- 1 tsLibGUI/tsWxPkg/src/tsWx.py (Available)
- 2 tsLibGUI/tsWxPkg/src/tsWxAcceleratorEntry.py (Available)
- 3 tsLibGUI/tsWxPkg/src/tsWxAcceleratorTable.py (Available)
- 4 tsLibGUI/tsWxPkg/src/tsWxApp.py (Available)
- 5 tsLibGUI/tsWxPkg/src/tsWxBoxSizer.py (Available)
- 6 tsLibGUI/tsWxPkg/src/tsWxButton.py (Available)

- 7** tsLibGUI/tsWxPkg/src/tsWxCallLater.py (Future)
- 8** tsLibGUI/tsWxPkg/src/tsWxCaret.py (Available)
- 9** tsLibGUI/tsWxPkg/src/tsWxCheckBox.py (Available)
- 10** tsLibGUI/tsWxPkg/src/tsWxChoice.py (Future)
- 11** tsLibGUI/tsWxPkg/src/tsWxColor.py (Available)
- 12** tsLibGUI/tsWxPkg/src/tsWxColorDatabase.py (Available)
- 13** tsLibGUI/tsWxPkg/src/tsWxCommandLineEnv.py (Available)
- 14** tsLibGUI/tsWxPkg/src/tsWxControl.py (Available)
- 15** tsLibGUI/tsWxPkg/src/tsWxControlWithItems.py (Available)
- 16** tsLibGUI/tsWxPkg/src/tsWxCursor.py (Available)
- 17** tsLibGUI/tsWxPkg/src/tsWxDebugHandlers.py (Future)
- 18** tsLibGUI/tsWxPkg/src/tsWxDialog.py (Available)
- 19** tsLibGUI/tsWxPkg/src/tsWxDialogButton.py (Available)
- 20** tsLibGUI/tsWxPkg/src/tsWxDisplay.py (Available)
- 21** tsLibGUI/tsWxPkg/src/tsWxDoubleLinkedList.py (Available)
- 22** tsLibGUI/tsWxPkg/src/tsWxEraseEvent.py (Future)
- 23** tsLibGUI/tsWxPkg/src/tsWxEvent.py (Available)
- 24** tsLibGUI/tsWxPkg/src/tsWxEventDaemon.py (Future)
- 25** tsLibGUI/tsWxPkg/src/tsWxEventLoop.py (Available)
- 26** tsLibGUI/tsWxPkg/src/tsWxEventLoopActivator.py (Future)
- 27** tsLibGUI/tsWxPkg/src/tsWxEventQueueEntry.py (Future)
- 28** tsLibGUI/tsWxPkg/src/tsWxEventTableEntry.py (Available)
- 29** tsLibGUI/tsWxPkg/src/tsWxEvtHandler.py (Available)
- 30** tsLibGUI/tsWxPkg/src/tsWxFlexGridSizer.py (Future)
- 31** tsLibGUI/tsWxPkg/src/tsWxFocusEvent.py (Future)
- 32** tsLibGUI/tsWxPkg/src/tsWxFrame.py (Available)
- 33** tsLibGUI/tsWxPkg/src/tsWxFrameButton.py (Available)
- 34** tsLibGUI/tsWxPkg/src/tsWxGauge.py (Available)
- 35** tsLibGUI/tsWxPkg/src/tsWxGlobals.py (Available)
- 36** tsLibGUI/tsWxPkg/src/tsWxGraphicalTextUserInterface.py (Available)
- 37** tsLibGUI/tsWxPkg/src/tsWxGridBagSizer.py (Future)
- 38** tsLibGUI/tsWxPkg/src/tsWxGridSizer.py (Available)
- 39** tsLibGUI/tsWxPkg/src/tsWxItemContainer.py (Future)
- 40** tsLibGUI/tsWxPkg/src/tsWxKeyboardState.py (Available)

- 41 tsLibGUI/tsWxPkg/src/tsWxKeyEvent.py (Available)
- 42 tsLibGUI/tsWxPkg/src/tsWxLayoutAlgorithm.py (Future)
- 43 tsLibGUI/tsWxPkg/src/tsWxListBox.py (Future)
- 44 tsLibGUI/tsWxPkg/src/tsWxLog.py (Future)
- 45 tsLibGUI/tsWxPkg/src/tsWxMenu.py (Future)
- 46 tsLibGUI/tsWxPkg/src/tsWxMenuBar.py (Future)
- 47 tsLibGUI/tsWxPkg/src/tsWxMouseEvent.py (Available)
- 48 tsLibGUI/tsWxPkg/src/tsWxMouseState.py (Available)
- 49 tsLibGUI/tsWxPkg/src/tsWxMultiFrameEnv.py (Available)
- 50 tsLibGUI/tsWxPkg/src/tsWxObject.py (Available)
- 51 tsLibGUI/tsWxPkg/src/tsWxPanel.py (Available)
- 52 tsLibGUI/tsWxPkg/src/tsWxPoint.py (Available)
- 53 tsLibGUI/tsWxPkg/src/tsWxPyApp.py (Available)
- 54 tsLibGUI/tsWxPkg/src/tsWxPyEventBinder.py (Available)
- 55 tsLibGUI/tsWxPkg/src/tsWxPyOnDemandOutputWindow.py (Available)
- 56 tsLibGUI/tsWxPkg/src/tsWxPySimpleApp.py (Available)
- 57 tsLibGUI/tsWxPkg/src/tsWxPySizer.py (Available)
- 58 tsLibGUI/tsWxPkg/src/tsWxRadioBox.py (Available)
- 59 tsLibGUI/tsWxPkg/src/tsWxRadioButton.py (Available)
- 60 tsLibGUI/tsWxPkg/src/tsWxRect.py (Available)
- 61 tsLibGUI/tsWxPkg/src/tsWxScreen.py (Available)
- 62 tsLibGUI/tsWxPkg/src/tsWxScrollBar.py (Available)
- 63 tsLibGUI/tsWxPkg/src/tsWxScrollBarButton.py (Available)
- 64 tsLibGUI/tsWxPkg/src/tsWxScrollBarGauge.py (Available)
- 65 tsLibGUI/tsWxPkg/src/tsWxScrolled.py (Available)
- 66 tsLibGUI/tsWxPkg/src/tsWxScrolledText.py (Available)
- 67 tsLibGUI/tsWxPkg/src/tsWxScrolledWindow.py (Available)
- 68 tsLibGUI/tsWxPkg/src/tsWxShowEvent.py (Available)
- 69 tsLibGUI/tsWxPkg/src/tsWxSize.py (Available)
- 70 tsLibGUI/tsWxPkg/src/tsWxSizer.py (Available)
- 71 tsLibGUI/tsWxPkg/src/tsWxSizerFlags.py (Available)
- 72 tsLibGUI/tsWxPkg/src/tsWxSizerItem.py (Available)
- 73 tsLibGUI/tsWxPkg/src/tsWxSizerItemList.py (Available)
- 74 tsLibGUI/tsWxPkg/src/tsWxSizerSpacer.py (Available)

- 75** tsLibGUI/tsWxPkg/src/tsWxSlider.py (Future)
- 76** tsLibGUI/tsWxPkg/src/tsWxSplashScreen.py (Available)
- 77** tsLibGUI/tsWxPkg/src/tsWxStaticBox.py (Available)
- 78** tsLibGUI/tsWxPkg/src/tsWxStaticBoxSizer.py (Available)
- 79** tsLibGUI/tsWxPkg/src/tsWxStaticLine.py (Available)
- 80** tsLibGUI/tsWxPkg/src/tsWxStaticText.py (Available)
- 81** tsLibGUI/tsWxPkg/src/tsWxStatusBar.py (Available)
- 82** tsLibGUI/tsWxPkg/src/tsWxSystemSettings.py (Available)
- 83** tsLibGUI/tsWxPkg/src/tsWxTaskBar.py (Available)
- 84** tsLibGUI/tsWxPkg/src/tsWxTextCtrl.py (Available)
- 85** tsLibGUI/tsWxPkg/src/tsWxTimer.py (Future)
- 86** tsLibGUI/tsWxPkg/src/tsWxToggleButton.py (Future)
- 87** tsLibGUI/tsWxPkg/src/tsWxTopLevelWindow.py (Available)
- 88** tsLibGUI/tsWxPkg/src/tsWxValidator.py (Future)
- 89** tsLibGUI/tsWxPkg/src/tsWxWindow.py (Available)

5 FIXED BUGS & ISSUES

- *Summary of Fixed Bugs & Issues* (on page 121)
- *Details of Fixed Bugs & Issues* (on page 125)

5.1 Summary of Fixed Bugs & Issues

The Public GitHub Repository contains the record of all source code and document file versions and associated repository user reported comments, open, closed and pending bugs and issues.

- ***Fixed Bugs and Issues available on Public GitHub Repository*** (on page 124) --- Production, Release Candidate, Beta, Alpha and Prototype phase version information.

Excerpt From Wikipedia, the free encyclopedia:

"GitHub is a web-based Git repository hosting service. It offers all of the distributed revision control and source code management (SCM) functionality of Git as well as adding its own features. Unlike Git, which is strictly a command-line tool, GitHub provides a Web-based graphical interface and desktop as well as mobile integration. It also provides access control and several collaboration features such as bug tracking, feature requests, task management, and wikis for every project."

GitHub offers both plans for private repositories and free accounts, which are usually used to host open-source software projects. As of April 2016, GitHub reports having more than 14 million users and more than 35 million repositories, making it the largest host of source code in the world."

Excerpt From Wikipedia, the free encyclopedia:

"In software engineering, software configuration management (SCM or S/W CM) is the task of tracking and controlling changes in the software, part of the larger cross-disciplinary field of configuration management. SCM practices include revision control and the establishment of baselines. If something goes wrong, SCM can determine what was changed and who changed it. If a configuration is working well, SCM can determine how to replicate it across many hosts.

The acronym "SCM" is also expanded as source configuration management process and software change and configuration management. However, "configuration" is generally understood to cover changes typically made by a system administrator."

- ***Fixed Bugs and Issues available on Private Bugzilla Server*** (on page 124) --- Alpha and Prototype phase version information.

Excerpt From Wikipedia, the free encyclopedia:

"Bugzilla is a web-based general-purpose bugtracker and testing tool originally developed and used by the Mozilla project, and licensed under the Mozilla Public License."

Released as open-source software by Netscape Communications in 1998, it has been adopted by a variety of organizations for use as a bug tracking system for both free and open-source software and proprietary projects and products. Bugzilla is used, among others, by the Mozilla Foundation, WebKit, Linux kernel, FreeBSD, GNOME, KDE, Apache, Red Hat, Eclipse and LibreOffice. It is also self-hosting."

- **Fixed Bugs and Issues available in Released Documents** (see "**Fixed Bugs and Issues available in Released Documents**" on page 122) --- Prototype phase version information.

5.1.1 Fixed Bugs and Issues available in Released Documents

DATE	ISSUE	DESCRIPTION
2012/11/27	The <i>TeamSTARS</i> "tsWxGTUI_PyVx" Toolkit needs to become one of the installed Python modules in order to become generally usable by application developers and system operators.	<p>Unless the <i>TeamSTARS</i> "tsWxGTUI_PyVx" Toolkit has been "installed" within the "site-packages" library associated with the default Python interpreter's run time library, application programs can only be run from the same "tsWxGTUI_PyVx" directory that contains the "tsLibCLI" and "tsLibGUI" directory.</p> <p>An install tool (setup.py) has been created using "distutils" technology.</p> <p>The process, for the default python, includes the following phases:</p> <ul style="list-style-type: none"> ▪ python setup.py build ▪ python setup.py install (NOTE: super user privileges are typically required)
2013/09/17	Python 3.0 introduced use of byte arrays for curses termname and longname methods. 2to3 conversion failed to automatically convert byte arrays to ascii strings.	<p>For Python-3x version only, added decoding logic to methods longname and termname.</p> <ul style="list-style-type: none"> ▪ reply = curses.longname().decode("ascii", "strict") ▪ reply = curses.termname().decode("ascii", "strict") <p>Without the conversion, the tsWxGraphicalTextUserInterface module would trap and report its inability to find a valid terminal type such as 'xterm' because curses reported a b'xterm'.</p> <p>The following logic did not resolve the problem:</p> <ul style="list-style-type: none"> ▪ reply = str(curses.termname()) ▪ reply = repr(curses.termname()) ▪ reply = ascii(curses.termname())

2013/12/01	SplashScreen Bitmap Image Compatibility	<p>Reported 2011/12/06:</p> <p>SplashScreen Bitmap image are platform dependant. An image created and saved on a Cygwin console will reload and be displayed only on a Cygwin console. It will not reload and display on a Linux xterm.</p> <p>Resolved 2013/12/01:</p> <p>The "tsWxGraphicalTextUserInterface" module has been re-engineered to create, name, save, reload and install a Bitmap image for each platform. It was necessary for the design to support the operator used screen column x row size and platform-specific operating system names. For example:</p> <ul style="list-style-type: none"> ▪ Base Name Size Type Host OS File Ext. ▪ ----- ▪ "SplashScreen-[60x15_CYGWIN]-cygwin_nt-5.0.bmp" ▪ "SplashScreen-[60x15_LINUX]-cygwin_nt-6.1.bmp" ▪ "SplashScreen-[60x15_XTERM]-cygwin_nt-6.2.bmp" ▪ "SplashScreen-[80x25_XTERM]-darwin.bmp" ▪ "SplashScreen-[128x50_XTERM]-freebsd.bmp" ▪ "SplashScreen-[128x50_XTERM]-sunos.bmp" ▪ "SplashScreen-[80x40_VT100]-cygwin_nt-5.0.bmp" ▪ "SplashScreen-[80x15_VT220]-cygwin_nt-6.3.bmp" <p>In addition, to isolate the "tsWxGraphicalTextUserInterface" module from user changes, the "tsWxGlobals" module has been enhanced to define the user modifiable splash screen configuration control options and default Python doc string data items for Trademark, Copyright, License and Notices sections.</p>
2010/12/15	Lack of mouse and/or/keyboard shortcut key support for nNon-olor vt100/vt220 terminals and emulators	<p>Resolved 2015/06/23.</p> <p>Module tsWxEventLoop in tsLibGUI was modified to decode mouse input data that did not conform to that for color xterm-family terminals and terminal emulators. Six mouse input events signals (character codes) were received every time a mouse button was pressed or released. A click therefore involved twelve mouse input event signals (character codes) which where then parsed to create the (mouseID, x, y, z, bstate [equivalent for a single click but not capable of detecting/representing double or triple clicks]) tuple to the xterm mouse input.</p>

2012/11/27	<p>The "python setup.py sdist" command to the distutils package, cannot be used to create a "tar" file or "zip" file for a repository containing:</p> <ul style="list-style-type: none"> Source code for both Python-2x and Python-3x Notebook for both Python-2x and Python-3x engineering documentation in application-specific multi-font formats. 	<p>Resolved 2015/05/25.</p> <ol style="list-style-type: none"> As a work around, the distribution now includes the following POSIX-style commnd line interface shell scripts that can be invoked only within the "<Top Level Recipient Repository>": <ul style="list-style-type: none"> setup_tsWxGTUI_PyVx_tar_file.sh setup_tsWxGTUI_PyVx_zip_file.sh For Python-2x, the "python2 setup.py build" / "python2 setup.py install" commands to the distutils package can be invoked only within the directory: <ul style="list-style-type: none"> "./Site-Packages/tsWxGTUI_PyVx/Python-2x" For Python-3x, the "python3 setup.py build" / "python3 setup.py install" commands to the distutils package can be invoked only within the directory: <ul style="list-style-type: none"> "./Site-Packages/tsWxGTUI_PyVx/Python-3x" .
------------	---	---

5.1.2 Fixed Bugs and Issues available on Private Bugzilla Server

DATE	ISSUE	DESCRIPTION
2015/04/23	<p>The text-file-based <i>TeamSTARS</i> "tsWxGTUI_PyVx" Toolkit fixed Bugs and Issues list has become too cumbersome to create, modify and search.</p>	<p>The contents of the <i>Fixed Bugs and Issues available in Released Documents</i> (see "<i>Fixed Bugs and Issues available in Released Documents</i>" on page 122) has been recreated on Bugzilla WEB server. It now permits one or more authorized authors and users to independently access, add, view and modify the registered private contents.</p> <p>This eliminates the need for text-file maintenance.</p>

5.1.3 Fixed Bugs and Issues available on Public GitHub Repository

DATE	ISSUE	DESCRIPTION
2015/07/24	<p>The Bugzilla WEB server-based <i>TeamSTARS</i> "tsWxGTUI_PyVx" Toolkit fixed Bugs and Issues list is NOT accessible to the general public after the GitHub release.</p>	<p>The contents of the <i>Fixed Bugs and Issues available on Private Bugzilla Server</i> (on page 124) has been recreated on the <i>TeamSTARS "tsWxGTUI_PyVx" Toolkit Repository</i> (https://github.com/rigordo959/tsWxGTUI_PyVx_Repository). It now permits one or more repository authors and users to independently access, add, view and modify the registered publicly visible contents.</p> <p>This eliminates the need for Bugzilla WEB server and text-file maintenance. It enables the public to register new issues and to submit comments on existing issues and comments.</p>

5.2 Details of Fixed Bugs & Issues

- *getOptions API* (on page 126)

Draft

5.2.1 getOptions API

2013/04/15 rsg Modify the Application Programming Interface (API) of the "tsLibCLI" and "tsLibGUI" Toolkit by removing the Command Line parsing method "getOptions" and associated references from the following:

"tsApplication" primitive module of the "tsLibCLI" Toolkit

"tsCommandLineEnv" convenience module of the "tsLibCLI" Toolkit

"tsWxMultiFrameEnv" convenience module of the "tsLibGUI" Toolkit

History of API's for Command Line parsing:

Python 1.4 introduced the "getopt" module. It is a parser for command line options whose API is designed to be familiar to users of the C getopt() function. It parses command line options and parameter list.

args is the argument list to be parsed, without the leading reference to the running program. Typically, this means sys.argv[1:].

options is the string of option letters that the script wants to recognize, with options that require an argument followed by a colon (':'); i.e., the same format that Unix getopt() uses).

Python 2.3 introduced the "optparse" module. It is a more convenient, flexible, and powerful library for parsing command-line options than the old "getopt" module. It uses a more declarative style of command-line parsing: one creates an instance of "OptionParser", populates it with options, and parse the command line. It allows you to specify options in the conventional GNU/POSIX syntax, and additionally generates usage and help messages for you.

Python 2.7 introduced the "argparse" module. It makes it easy to write user-friendly command-line interfaces. The program defines what arguments it re-

quires, and "argparse" will figure out how to parse those out of sys.argv. It also automatically generates help and usage messages and issues errors when users give the program invalid arguments.

Rational:

Command Line parsing is within the scope of responsibility of the application, NOT the toolkit which only provides building block services for the application to use when appropriate.

Preserving the "getOptions" API is no longer practical because the "argparse" module is NOT backward compatible with its predecessor, the "optparse" module. The incompatibility is illustrated in the following code fragments and results (re-formatted to fit within the 60-column width of this document).

```

---- OptParse Test Code -----

from optparse import OptionParser

print('\n    %s' % 'optparse')

parser = OptionParser()
parser.add_option(
    "-f", "--file", dest="filename",
    help="write report to FILE", metavar="FILE")
parser.add_option(
    "-q", "--quiet",
    action="store_false", dest="verbose", default=True,
    help="don't print status messages to stdout")

(options, args) = parser.parse_args()
print('\n\toptions=%s; type=%s' % (
    str(options), str(type(options))))

print('\n\targs=%s; type=%s' % (str(args),
    str(type(args))))

---- OptParse Test Results # 1 -----

$ python sample.py -h

    optparse
Usage: sample.py [options]

Options:
  -h, --help            show this help message and exit

```

```

    -f FILE, --file=FILE  write report to FILE
    -q, --quiet           don't print status messages
                           to stdout

---- OptParse Test Results # 2 -----

$ python sample.py

    optparse

        options={'verbose': True, 'filename': None};
                type=<type 'instance'>

        args=[]; type=<type 'list'>

---- OptParse Test Results # 3 -----

$ python sample.py -q 100 200

        options={'verbose': False,
                  'filename': None};
                type=<type 'instance'>

        args=['100', '200']; type=<type 'list'>

---- ArgParse Test Code -----

import argparse

print('\n    %s' % 'argparse')

parser = argparse.ArgumentParser(
    description='Process some integers.')
parser.add_argument(
    'integers', metavar='N', type=int, nargs='+',
    help='an integer for the accumulator')
parser.add_argument(
    '--sum', dest='accumulate', action='store_const',
    const=sum, default=max, help='sum the integers (
    default: find the max)')

args = parser.parse_args()
print('\nargs=%s;\ntype=%s' % (str(args),
    str(type(args))))

print('\n\targs.accumulate(args.integers)=%s' % \
    str(args.accumulate(args.integers)))

print('\n\targs.integers=%s' % str(args.integers))

---- ArgParse Test Results # 1 -----

$ python sample.py -h

    argparse
usage: sample.py [-h] [--sum] N [N ...]

```

Process some integers.

positional arguments:

 N an integer for the accumulator

optional arguments:

 -h, --help show this help message and exit
 --sum sum the integers (default: find the max)

---- ArgParse Test Results # 2 -----

\$ python sample.py

```
    argparse
usage: sample.py [-h] [--sum] N [N ...]
sample.py: error: too few arguments
```

```
    print('\n\toptions=%s; type=%s' % (str(options),
                                       str(type(options))))
```

```
    print('\n\targs=%s; type=%s' % (str(args),
                                     str(type(args))))
```

---- ArgParse Test Results # 3 -----

\$ python sample.py 100 200 300 400

```
    argparse

args=Namespace(accumulate=<built-in function max>,
               integers=[100, 200, 300, 400]);
type=<class 'argparse.Namespace'>
```

```
        args.accumulate(args.integers)=400
```

```
        args.integers=[100, 200, 300, 400]
```

---- ArgParse Test Results # 4 -----

\$ python sample.py --sum 100 200 300 400

```
    argparse

args=Namespace(accumulate=<built-in function sum>,
               integers=[100, 200, 300, 400]);
type=<class 'argparse.Namespace'>
```

```
        args.accumulate(args.integers)=1000
```

```
        args.integers=[100, 200, 300, 400]
```

6 KNOWN BUGS & ISSUES

- *Design Issues* (on page 130)
- *Display Issues* (on page 137)
- *Import Issues* (on page 138)
- *Installation Issues* (on page 139)
- *Operational Issues* (on page 139)
- *Remote Access Issues* (on page 139)
- *Troubleshooting Issues* (on page 140)
- *User Input Issues* (on page 147)

6.1 Design Issues

DATE	ISSUE	DESCRIPTION
2011/11/10	Application programs cannot delete wxPython-style GUI objects.	<p>Application programs cannot delete wxPython-style GUI objects because Python's "nCurses" module does not have the "nCurses_delwin" method for deleting "nCurses" style GUI objects.</p> <p>Not sure this issue can be resolved. A search for an explanation for why Python does not support the delwin function, came upon the following "nCurses" man page entry:</p> <ul style="list-style-type: none">▪ Calling delwin deletes the named window, freeing all memory associated with it (it does not actually erase the window's screen image). Subwindows must be deleted before the main window can be deleted.
2011/11/10	Abort Signal (Ctrl-C) Handling	<p>Applications occasionally terminate without restoring the display to its previous state in which keyboard input is echoed to the display.</p> <p>Troubleshooting Hint:</p> <ul style="list-style-type: none">▪ The operator must type "stty sane" to manually restore the display to its normal state.▪ The operator might want to Clear Screen by typing Ctrl-L shell command "^L".
2012/07/03	Conflict between need for wxPython's "Show" method and emulator's need for nCurses' Panel Library	<p>Limited screen real estate typically necessitates the overlapping of window objects. When the operator needs to see the features of a partially or totally hidden object, a Graphical User Interface uses a Task Bar with buttons for each frame or dialog. After moving the mouse over the button</p>

		<p>for the hidden object, the operator can click on the mouse button to bring a hidden background object to the visible foreground. The nCurses-based design provides a similar Task Bar.</p> <p>However, the current design's use of the wxWidgets-/wxPython-style "Show" method is incompatible with use of the nCurses Panel (Stack) library as evidenced by sample code (See http://tldp.org/HOWTO/NCURSES-Programming-HOWTO/panels.html). The sample code doesn't use components associated with "Show" method, but the wxPython emulation depends on them to accomodate a delay in the creation of nCurses windows until after the completion of associated distributed layouts.</p>
2012/08/11	Lack of application program blocking mechanism for keyboard input.	<p>Keyboard input subject to restrictions:</p> <ul style="list-style-type: none"> ▪ tsWxPasswordEntryDialog is derived from tsWxTextEntryDialog. The latter uses tsWxTextEditBox which has usability issues in both non-event driven and event driven modes. ▪ The non-event-driven mode uses curses.textpad.Textbox which monopolizes GUI platform and ignores mouse button clicks until the operator terminates input via double entry of control-G key. It also, responds to operator input after the control-G. ▪ The event driven mode uses an python emulation of curses.textpad.Textbox that provides enhanced edit capabilities while recognizing mouse button clicks and keyboard press-release activity. However, the display blinks as each new text character is displayed. More importantly, there is no mechanism to block the application program pending the operator's completion of text input.
2012/11/16	Mouse "Click" Event handling is unreliable.	<p>Mouse "Click" Event handling is currently unreliable. What works on Windows with Cygwin sometimes crashes on Ubuntu Linux.</p> <p>Troubleshooting Hint:</p> <ul style="list-style-type: none"> ▪ Detected caret position, at time of mouse click, considered to be outside of displayed GUI object border. ▪ Perhaps displayed GUI object border must be pre-aligned with character cell display.
2012/11/16	Lack of Pending Event handling.	<p>wxPython-style event handling should include front-end (real time) dispatching or queuing with back-end (idle time) ProcessPendingEvent handling.</p> <ul style="list-style-type: none"> ▪ While the wxPython emulation is under construction, the workaround involves front-end use of tsWxProcessSelectedEventTable. ▪ The workaround has been used after much trial and error to demonstrate buttons, checkboxes, radio buttons and the scroll bar "arrow" buttons and gauges associated with text

		scrolling.
2013/05/09	Xemacs Syntax Error: "GUI item produces too long displayable string"	<p>Xemacs reports syntax error when editing tsApplication and tsCommandLineParser modules.</p> <ul style="list-style-type: none"> ▪ Xemacs unable to create list of functions in the file being edited and without the list one can only use search to find a the function entry point. ▪ Xemacs takes minutes to format the file for viewing after edit changes. ▪ Refactoring module to create separate files for each function restores normal Xemacs capabilities but requires modifying the function to simulate the class member behavior. Modifications include: a) import module as function; b) self.function = function; c) explicitly referencing function as self.function(self)
2013/07/18	Need Frame and Dialog control button event handlers	<ul style="list-style-type: none"> ▪ Frame control buttons (ICONIZE, MAXIMIZE/RESTOREDOW, and CLOSE) do not make appropriate changes to display. The handlers only output an appropriate text message. ▪ Dialog control buttons (HELP and CLOSE) do not make appropriate changes to display. The handlers only output an appropriate text message.
2013/08/16	Hierarchical library file structure interferes with operation of pylint and pydoc	<p>Pylint is a tool that checks for errors in Python code, tries to enforce a coding standard and looks for bad code smells. This is similar but nevertheless different from what pychecker provides, especially since pychecker explicitly does not bother with coding style. The default coding style used by Pylint is close to PEP 008 (aka Guido's style guide).</p> <p>Pydoc is a documentation module for the programming language Python. Similar to the functionality of Perldoc within Perl, Pydoc allows Python programmers to access Python's documentation help files, generate HTML pages with documentation specifics, and find the appropriate module for a particular job.[1] Pydoc can be accessed from a module-specific GUI,[2] from within the Python interpreter, or from a command line shell.[1][3]</p> <p>Pylint and Pydoc expect to process files from within a monolithic file directory. When invoked within a hierarchical file directory with its hierarichal "package" import mechanism, they report various import errors which degrade their usefulness.</p> <ul style="list-style-type: none"> ▪ "tsWxGTUI" uses a hierarchical file system to organize and segregate building blocks, tests and tools associated with the Command Line Interface (CLI) from building blocks, tests and tools associated with the Graphical-style User Interface (GUI). <hr/> <p>Segregation prevented CLI building blocks from unintentionally becoming dependent on GUI ones.</p> <p>However, it did not prevent GUI building blocks from applying CLI ones when appropriate.</p>

		<ul style="list-style-type: none"> Before the segregation of tsLibCLI from tsLibGUI, there was a single directory, tsLibraries. Replicating the tsLibCLI files in the tsLibGUI directory seemed to resolve the pylint issue. After the segregation of tsLibCLI from tsLibGUI, individual files imported the associated tsLibCLI or tsLibGUI before importing their associated building blocks. This caused the error reports and interfered with the operation of pylint and pydoc. During development, it seems more important to focus on "tsWxGTUI" Toolkit implementation issues that facilitate operation and troubleshooting than on source code checking and documenting. This decision should be reconsidered before any non-developmental product release.
2013/08/20	Python files must be "untabified" before being processed by tsStripComments to avoid "IndentationError".	<p>Failure to "untabify" Python source files prior to processing by "tsStripComments" may need manual fix-up of IndentationError.</p> <ul style="list-style-type: none"> Launch of test_tsOperatorSettingsParser failed: <pre> Importing tsLibCLI.tsCommandLineEnvPkg.src.tsCommandLi neEnv Traceback (most recent call last): File "test_tsOperatorSettingsParser.py", line 33, in <module> import tsLibCLI File "/cygdrive/d/Temp/WithoutComments/For_JimWilk inson/tsLibCLI/__init__.py", line 51, in <module> __import__(fullModuleName) File "/cygdrive/d/Temp/WithoutComments/For_JimWilk inson/tsLibCLI/tsCommandLineEnvPkg/src/__init_ _.py", line 52, in <module> __import__(fullModuleName) File "/cygdrive/d/Temp/WithoutComments/For_JimWilk inson/tsLibCLI/tsCommandLineEnvPkg/src/tsCom mandLineEnv.py", line 72 print("\n\n\tCommandLineEnv.Wrapper') ^ IndentationError: unindent does not match any outer indentation level </pre>

		<ul style="list-style-type: none"> Upon investigation, determined that the three print statements in the original source file had NOT been "untabified" (i.e., they had both tabs and blanks)
2013/08/24	tsStripComments renders unusable such doc string dependent modules as tsOperatorSettingsParser	<p>Blank lines within doc strings are visually used to define the end of one paragraph and the start of a new one.</p> <ul style="list-style-type: none"> tsOperatorSettingsParser uses the text wrap module to ensure that text fits within the available run time display area. Text wrap depends on blank lines to avoid merging text from separate paragraphs. Authoring doc strings with blank lines is both natural and effective at achieving the correct display. tsStripComments originally did not recognize the appropriateness of removing some blank lines but not others. An update removed all but the first of a sequence of consecutive blank lines. The main application using a tsOperatorSettingsParser-like module passes the module's launch purpose via a doc string which is subject to stripping. All other launch parameters (such as title, author, build version and build date) are passed by reference to the associated labeled doc string. See BuildPurpose AttributeError (on page 136) for traceback example when doc string deleted and Purpose = __doc__ attempted to pass a NoneType. A simple work around would be to create a purpose labeled doc string which will not be subject to comment stripping. A reference to the label can then be passed instead of the one to the unlabeled doc string.
2013/09/04	test_tsWxGridSizer event processing generates too many event messages per button click.	See Multiple GridSizer Mouse Click Events (on page 136)
2013/09/04	Partially-compliant wxPython User Interface. Keyboard input subject to restrictions	Text & Password Entry Dialogs (on page 137)
2013/09/06	tsConfigObjectPkg only available for Python-2.x.	<p>Python 3.x version of tsLib/tsConfObjectPkg/ reports TypeError in configobj.py line 1380, in _iced because: 'int' object "line[-1]" is not subscripted</p> <p>The source version, 4.7.2, was designed for Python 2.x. Its author, Michael Foord, has not ported in to Python 3.x. He "blessed" a port created by Zubin Mithra and Prashant Kuma but it could not be found on the referenced wiki and the 2to3 tool is unable to produce a workable version.</p>

[illegible]

6.1.2 BuildPurpose AttributeError

```
Traceback (most recent call last):
  File
"/cygdrive/d/Temp/WithoutComments/tsLibCLI/tsCommandLineEnvPkg/src/tsCommandLi
neEnv.py", line 114, in Wrapper
    self.runTimeEntryPoint()
  File "purposeless_unstripped_tsStripComments.py", line 1265, in EntryPoint
    (args, options) = getSettings()
  File "purposeless_unstripped_tsStripComments.py", line 1214, in getSettings
    (args, options) = myParser.parseCommandLineDispatch()
  File
"/cygdrive/d/Temp/WithoutComments/tsToolsCLI/tsStripCommentsPkg/src/tsStripSet
tingsParser.py", line 1091, in parseCommandLineDispatch
    (args, options) = self.parseCommandLineViaArgParse()
  File
"/cygdrive/d/Temp/WithoutComments/tsToolsCLI/tsStripCommentsPkg/src/tsStripSet
tingsParser.py", line 1278, in parseCommandLineViaArgParse
    self.parseCommandLineUsageViaArgParse()).replace(
  File
"/cygdrive/d/Temp/WithoutComments/tsToolsCLI/tsStripCommentsPkg/src/tsStripSet
tingsParser.py", line 1140, in parseCommandLineUsageViaArgParse
    lines = self.parent.buildPurpose.split('\n')
AttributeError: 'NoneType' object has no attribute 'split'
```

6.1.3 Multiple GridSizer Mouse Click Events

2013/09/04

test_tsWxGridSizer event processing generates too many event messages per button click. Button click event handling needs to be directly coupled to keypad button window rather than coupled to the parent Frame provided handler.

6.1.4 Text & Password Entry Dialogs

2013/09/04

Partially-compliant wxPython User Interface. Keyboard input subject to restrictions:

tsWxPasswordEntryDialog is derived from tsWxTextEntryDialog. The latter uses tsWxTextEditBox which has usability issues in both non-event driven and event driven modes.

The non-event-driven mode uses curses.textpad.Textbox which monopolizes GUI platform and ignores mouse button clicks until the operator terminates input via double entry of control-G key. It also, responds to operator input after the control-G.

The event driven mode uses an python emulation of curses.textpad.Textbox that provides enhanced edit capabilities while recognizing mouse button clicks and keyboard press-release activity. However, the display blinks as each new text character is displayed. More importantly, there is no mechanism to block the application program pending the operator's completion of text input.

6.2 Display Issues

DATE	ISSUE	DESCRIPTION
2011/12/06	GUI Object Automatic Layout Anomalies	<p>Use of wxPython-style sizers to automatically layout complex assemblies of GUI Objects typically produce the telescoping or overlapping of adjacent border lines into a single line.</p> <p>Conclusion: Use of wxPython-style pixel dimensions are associated with the nearest available nCurses column and row. Anomalies show up only when derived pixel dimensions are not integer multiples of the pixel width and height for the fixed width font of a single character. See an example at wxStaticBoxSizer.</p>
2013/03/05	Ubuntu Linux 12.04 terminal those runs "tsWxGTUI" applications associated with regression testing. Occassionally, a mouse click triggers an application trap for what appears to be an unknown cursor mis-alignment issue.	<p>Test Platforms:</p> <ul style="list-style-type: none"> Linux Ubuntu 12.04 32-bit Mac OS X (10.7.5) 64-bit Windows XP 32-bit (with cygwin, Unix-type environment) Windows 7 Professional 32-bit (with cygwin, Unix-type environment) Windows 8 Professional 32-bit (with cygwin, Unix-type environment) <p>Command Line Interface regression tests (Python 2.x and 3.x) include the following:</p>

		<ul style="list-style-type: none"> ▪ test_tsApplication.py ▪ test_tsCommandLineInterface.py ▪ test_tsDecorators.py ▪ test_tsExceptions.py ▪ test_tsLogger.py ▪ test_tsReportUtilities.py ▪ test_tsThreadPool.py ▪ tsLinesOfCode.py <p>Graphical User Interface regression tests (Python 2.x and 3.x) include the following:</p> <ul style="list-style-type: none"> ▪ test_tsWxBoxSizer.py ▪ test_tsWxGridSizer.py ▪ test_tsWxMetrics.py ▪ test_tsWxScrolledWindow.py (mouse click trap symptoms) ▪ test_tsWxWidgets.py
--	--	---

6.3 Import Issues

DATE	ISSUE	DESCRIPTION
2011/06/01	Non-compliant wxPython Library Modules	<p>Functions, Class Methods and Class Properties:</p> <ul style="list-style-type: none"> ▪ See Microsoft Access file: /sandbox-hg/Projects/tsWxPython.mdb
2011/06/01	Non-compliant wxPython Applications	<p>Unit, Integration and System Test Applications:</p> <ul style="list-style-type: none"> ▪ test_tsWxWidgets - Runs without failure. Unit Test for widgets. ▪ test_tsWxPySimpleApp - Runs without failure. Unit Test for startup.
2013/03/05	Named import module not found	<p>Despite establishment of tsLibCLI and tsLibGUI, importing within repository from test directories may fail.</p> <ul style="list-style-type: none"> ▪ Problem can be eliminated by installing "tsWxGTUI" into default Python's site-packages using the appropriate setup.py tool.

6.4 Operational Issues

6.5 Installation Issues

DATE	ISSUE	DESCRIPTION
2013/03/25	Access to the "tsToolsCLI" services should be path independant from the top level of the "tsWxGTUI" Toolkit site package.	<p>Access to the "tsToolsCLI" services is only via their source code path. Access should also be path independant from the top level of the "tsWxGTUI" Toolkit site package.</p> <ul style="list-style-type: none"> tsLibraryImport tsLinesOfCode tsPlatformQuery tsPublish tsStripComments tsStripLineNumbers tsTreeCopy tsTreeTrimLines
2013/03/25	Access to the "tsToolsGUI" services should be path independant from the top level of the "tsWxGTUI" Toolkit site package.	<p>Access to the "tsToolsGUI" services is only via their source code path. Access should also be path independant from the top level of the "tsWxGTUI" Toolkit site package.</p> <ul style="list-style-type: none"> tsWxLinesOfCode (Future) tsWxPlatformQuery (Future)

6.6 Remote Access Issues

DATE	ISSUE	DESCRIPTION
2012/04/07	A local platform connected to one or more remote platforms, which together are operating in "Stand-Among Mode", may report various connection errors.	<p>Use of the OpenSSH SSH client (remote login program) via the command "ssh <user id>@<remote host id>" may require the System Administrators, Software Engineers or System Operators to temporarily supend or permanently modify local and/or remote computer security settings:</p> <ul style="list-style-type: none"> Authorize local computer on remote system to resolve the issue: "ssh: connect to host <IP Address> port 22; Connection refused" Unblock firewall on remote system to resolve the issue: "sh: connect to host <IP Address> port 22; Connection"

		timed out"
--	--	------------

6.7 Troubleshooting Issues

DATE	ISSUE	DESCRIPTION
2013/09/04	Instrumentation to facilitate debugging has been left activated.	<i>Debugging Instrumentation</i> (on page 146)
2013/09/04	Development notes ("comments" and "doc strings") are embedded within each Python source file.	<i>Development Notes</i> (on page 147)

6.7.1 Wing IDE Configuration

Correspondence with Wing Support
<p>February 6, 2015 at 1:09AM</p> <p>I have not been able to use Wing 5.1 to debug Python curses application on Mac OS X or Cygwin under Windows 7. The last time I was able to do that was with Wing 3.</p> <p>The available how to has not been of much help. It lacks detailed examples.</p> <p>Richard S Gordon</p>
<p>February 6, 2015 at 10:40 AM</p> <p><i>On OS X, you need to use an external console; you set this up via the Options menu in the Debug I/O tool.</i></p> <p><i>Cygwin is more difficult to use because cygwin is effectively its own platform even though it runs on Windows.</i></p> <p><i>I suggest you get things working on OS X and then we can work on getting Cygwin to work.</i></p> <p><i>Wing IDE Support</i></p>
<p>February 6, 2015 at 7:20 PM</p> <p>There is not enough information on-line to get an external console working on OS X (Yosemite). The Mac OS X Terminal support curses by not a mouse The third-party iTerm supports curses and a mouse.external console did not launch any external console. Manually launching iTerm did not result in its use by Wing.</p> <p>Richard S Gordon</p>
<p>February 6, 2015 at 7:57 PM</p> <p>Follow-up: I Installed Wing 5.1 Pro on Ubuntu Linux 14.04 64-bit and set debug options as on Mac OS X. Reducing size of Wing desktop revealed Linux GDK console pop-up. Successfully ran Python-based CLI application and then a Curses-based Python application.</p> <p>Will try same thing on Mac OS X and let you know how it worked.</p>

Correspondence with Wing Support

Richard S Gordon

February 9, 2015 at 5:17 AM

I'm glad you got it working on Ubuntu.

On the Mac, you'd have to set up the Debugger > I/O > External Consoles preference so the first item launches iTerm. Currently the first item launches Terminal. This is done via resources/osx/run-in-terminal.applescript and you would need to write something similar for iTerm.

If you do that and want to send it to us, we'd be happy to add it to the set of defaults. I think we also need to make the External Consoles preference easier to deal with -- I see it doesn't allow rearranging items, which it should.

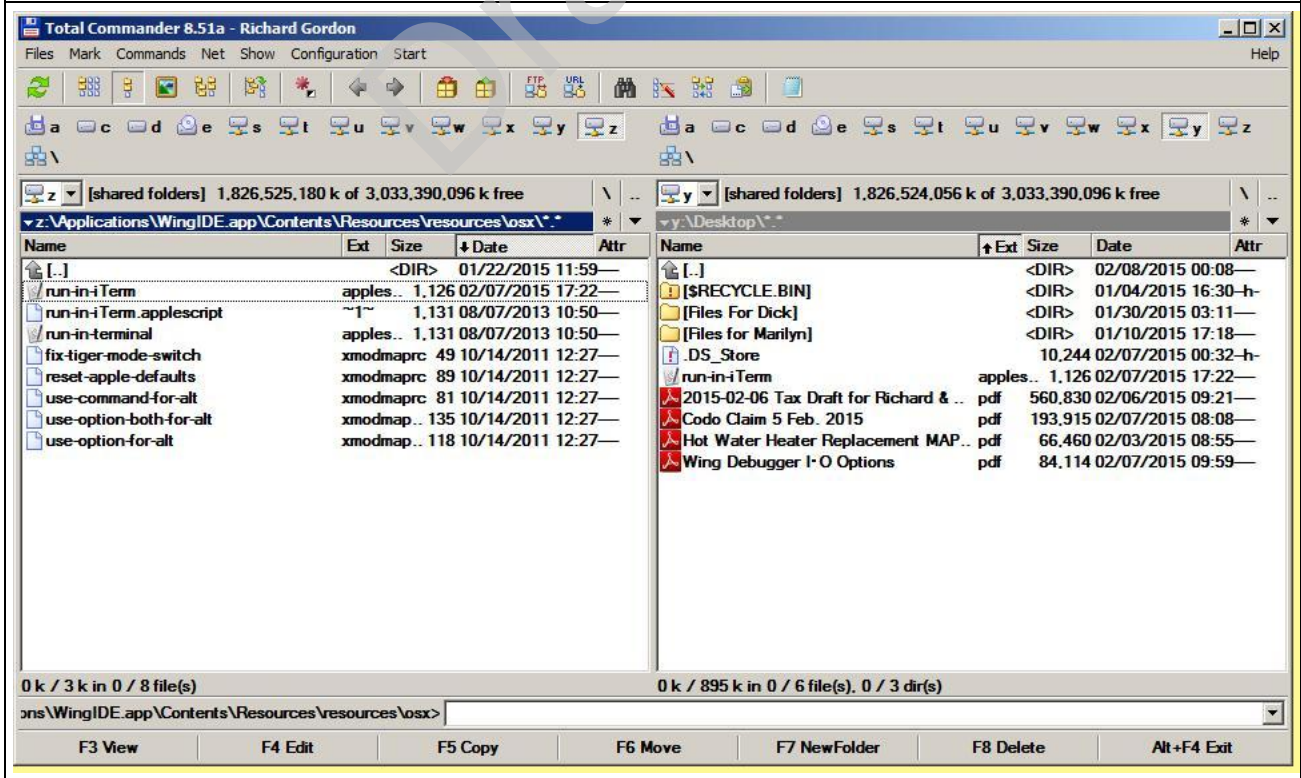
Wing IDE Support

February 8, 2015 at 12:47 AM

Attached is apple script for iTerm and screenshot of Debug I/O Options. Contrary to the screenshot, the Use External Console and External Console Waits on Exit were actually selected. The Mac Preview tool didn't capture my selections. I haven't figured out how to set terminal size (80 col. x 24 row) other than by dragging bottom right corner immediately after launch. The iTerm preferences (I typically use 80 col. x 50 row) only apply with a manual (non-scripted) launch.

In addition to support for 8-color xterm (default for Python 2.4-2.6), I also recommend support for xterm-16color (default for xterm-88color and xterm-256color (latter is default for Yosemite and Python 2.7.9 and 3.4.2)) and non-color vt100 and vt220. Linux platforms now offer a linux terminal alternative to 8-color xterm. Cygwin platforms offer a mouseless 8-color cygwin (mintty) terminal alternative to 8-color xterm.

Richard S Gordon



Correspondence with Wing Support

```
#!/usr/bin/osascript
run argv

    set cwd to do shell script "pwd"

    set cmd to "cd " & quoted form of cwd & ";"
    set i to 0
    repeat with arg in argv
        # Ignore any initial -e argument (arg = "-e" doesn't work for some
        # reason)
        if i = 0 and arg contains "-e" then
            # do nothing
        else
            set arg to quoted form of arg
            set cmd to cmd & " " & arg
        end if

        set i to i + 1
    end repeat

    set console_tab to my find_console_tab()

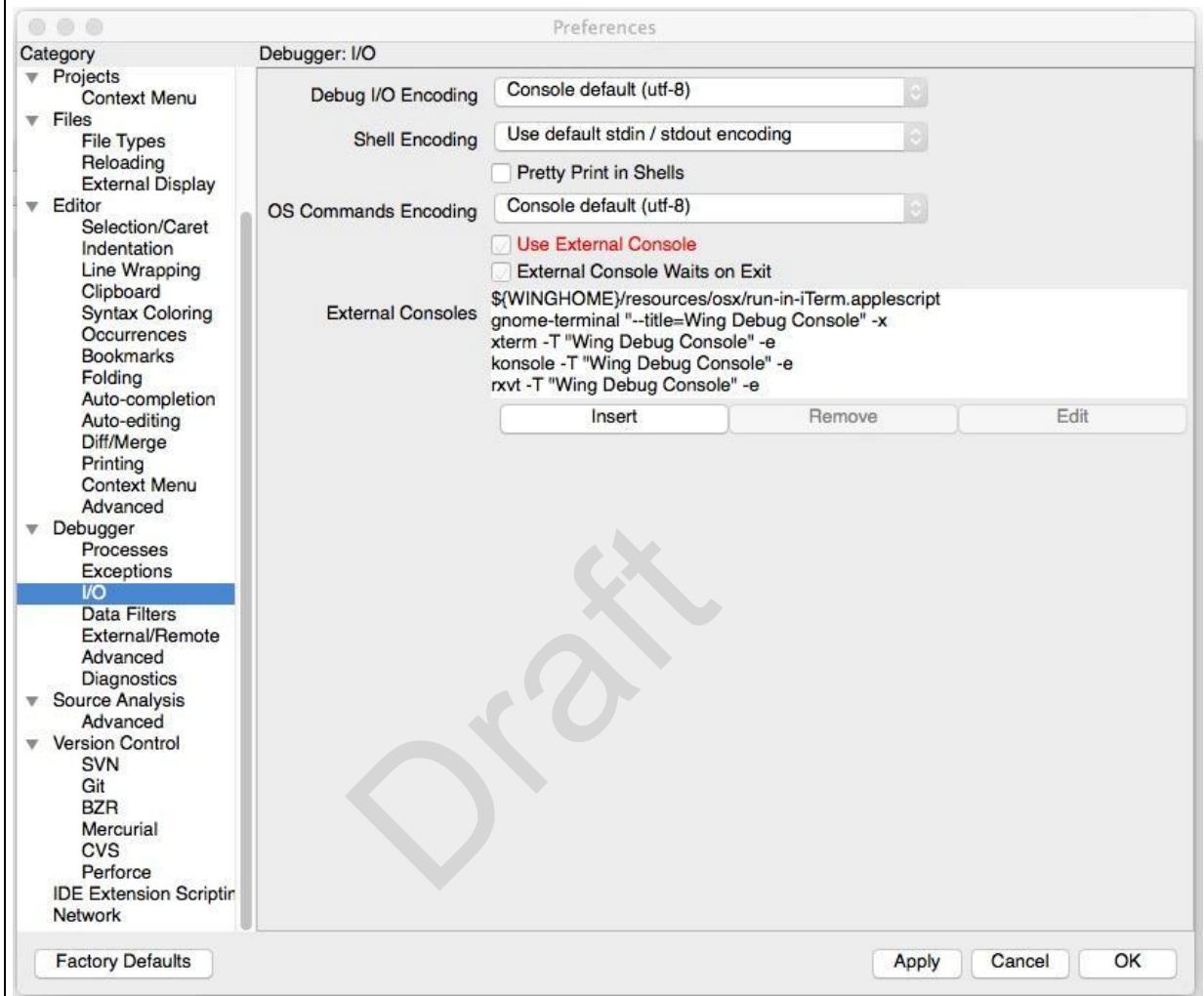
    tell application "iTerm"
        activate
        # do script cmd in console_tab
    end tell
end run
find_console_tab()
    tell application "iTerm"
        repeat with win in every window
            try
                set tab_list to every tab of win
            on error errmsg
                set tab_list to missing value
            end try

            if tab_list is not equal to missing value then
                repeat with t in tab_list

                    if (custom title of t) contains "Wing Debug
Console" then

                        if not (busy of t) then
                            return t
                        end if
                    end if
                end repeat
            end if
        end repeat

        set new_window to do script
        set custom title of new_window to "Wing Debug Console"
        return new_window
    end tell
end find_console_tab
```


Correspondence with Wing Support**February 9, 2015 at 5:17 AM**

CORRECTION: How to configure Wing 5.1 Pro for use with Python Curses on an external console

I have been unable to set size preferences for external console. I noticed that X11 was launched during launch of the Python application I was debugging. After uninstalling XQuartz-2.7.7, Wing would no longer launch my Python application. After re-installing XQuartz-2.7.7, Wing was able to launch my Python application. I apparently had overlooked one of the experiments I had tried to replicate my previously working Wing 3 configuration. would be great if Wing 5 did not need XQuartz as a prerequisite. But, as long as it works, Wing should state that curses/ncurses users will need XQuartz.

Dick Gordon

Correspondence with Wing Support

February 10, 2015 at 10:22 AM

Richard S. Gordon wrote:

I don't think neither iTerm or Terminal do anything. Only the X11 is needed.

In that case I'm confused...

X11 xterm is going to require XQuartz to be present to work because it's an X11 app.

you're trying to use iTerm as an alternative to xterm then -- assuming it's working, which it isn't for me -- it should no longer require XQuartz.

, the run-in-terminal.applescript default preference that ships with Wing (also a replacement for xterm) works without using XQuartz.

there's no way to run xterm without XQuartz...

February 12, 2015 at 7:30 AM

Sorry for the delayed response. I've had trouble formulating an answer that clearly captures the issues. Retesting various Linux, Mac OS X and Microsoft Windows platforms is time-consuming and is creating a complex spreadsheet with foot notes.

the interest of simplifying and clarifying our discussion, I propose to focus only on Mac OS X.

Wing 5.1.1 supports the internal and external Debug I/O console for Python applications which do not use the Python Curses module.

The Terminal utility supplied with Mac OS X supports Python applications which do not use the Python Curses module.

The third-party iTerm utility for Mac OS X supports Python applications which do not use the Python Curses module.

The third-party XQuartz X11 Terminal utility for Mac OS X supports Python applications which do not use the Python Curses module.

Only if third-party XQuartz has been installed on Mac OS X, Wing 5.1.1 supports the external Debug I/O console for Python applications which do use the Python Curses module.

Only if third-party XQuartz has not been installed on Mac OS X, Wing 5.1.1 fails to support the internal and external Debug I/O console for Python applications which do use the Python Curses module. The reported error is that Python Curses cannot find the TermInfo data base.

The Terminal utility supplied with Mac OS X supports Python applications which do use the Python Curses module but does not support mouse input.

The third-party iTerm for Mac OS X supports Python applications which do use the Python Curses module and does support mouse input.

The third-party XQuartz X11 Terminal utility for Mac OS X supports Python applications which do use the Python Curses module and does support mouse input.

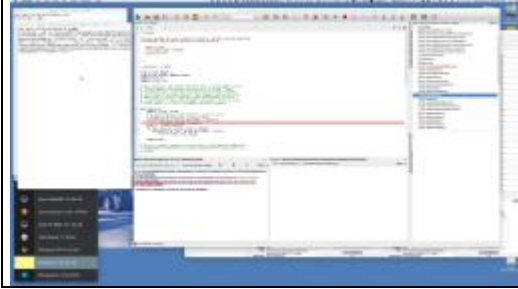
Dick Gordon

February 12, 2015 at 7:48 AM

You might find the attached screen shot useful.

Dick Gordon

Correspondence with Wing Support



Draft

6.7.2 Debugging Instrumentation

2013/09/04

Instrumentation to facilitate debugging has been left activated. Module level variables have been set to enable built-in code to log and display both normal progress and unexpected events.

The "tsWxGTUI" Toolkit is a work in progress. Unless one is introducing only localized changes, remembering where to find and turn debug control flags "on" and "off" quickly become tedious and vexing. Ignoring the automatically generated information until debugging is required is relatively painless.

Consider the lines-of-code metrics and debugging effort associated with just the primary Python version 2.x files (including documentation, tests and associated data but excluding their derived Python 3.x counterparts):

Overall Source Code Feature Statistics

	FILES	CODE	CMNTS	LINES	WORDS	CHARS
Pct:		48.51%	51.49%	100.00%		
Totals:	799	211904	224945	436849	1325892	16512687
Std:	722	399	450	795	2431	38612
Avg:	1	265	282	547	1659	20667

Distribution of Source Code Feature Statistics by File Types

TYPES	FILES	CODE	CMNTS	LINES	WORDS	CHARS	%-LINES
.ada	4	904	176	1080	4604	33596	0.25%
.adb	9	701	387	1088	3814	32307	0.25%
.ads	5	148	202	350	1886	14552	0.08%
.asm	15	355	452	807	3032	17602	0.18%
.bas	8	2160	8	2168	7184	46828	0.50%
.bash	4	24	140	164	772	4600	0.04%
.bat	4	16	8	24	32	204	0.01%
.c	22	6185	5117	11302	32960	262430	2.59%
.cpp	8	3712	1704	5416	14532	159776	1.24%
.f	8	2380	464	2844	9308	85340	0.65%
.f77	4	920	348	1268	4208	29836	0.29%
.f90	48	7620	5683	13303	48953	419506	3.05%
.ftn	4	236	168	404	1548	10160	0.09%
.h	10	307	223	530	1734	15481	0.12%
.inc	12	696	2292	2988	11708	133320	0.68%
.pas	23	4416	3782	8198	29599	231170	1.88%
.plm	8	3436	2980	6416	24768	191336	1.47%
.py	562	176695	179318	356013	1042775	12263799	81.50%
.sh	12	993	315	1308	5351	60881	0.30%
.txt	28	0	20372	20372	74999	2479155	4.66%
.y	1	0	806	806	2125	20808	0.18%

6.7.3 Development Notes

2013/09/04

Development notes ("comments" and "doc strings") are embedded within each Python source file.

Prior to publication the notes must be reviewed and edited to ensure accuracy, completeness and clarity.

For a pre-release said notes should be stripped out of each source file to be published. The "tsStripComments.py" tool is provided for this purpose. Even after note stripping, all executable statements (including imports, definitions, logic and input/output) will be fully human readable.

The "tsStripComments.py" tool must be used with care on those top-level application or test files which provide an operator with command line help. This is because general purpose command line parsers, such as "tsOperatorSettingsParser", expect to receive an application-specific purpose-defining doc string, text contained within triple quotes (such as `"""Module Purpose"""` or `'''Module Purpose'''`) that must not be stripped. The purpose doc string is typically unlabeled at or near the beginning of the file. It is referenced by `"__doc__"`. To ensure that it is not stripped, it is suggested that the `"__doc__"` label be explicitly defined via one of the following:

```
# Implicit label definition override
__doc__ = """Module Purpose"""
__doc__ = '''Module Purpose'''

# Explicit new label definition
purpose = """Module Purpose"""
purpose = '''Module Purpose'''
```

6.8 User Input Issues

DATE	ISSUE	DESCRIPTION
2008/09/03	Manually Sized Curses Screen	<p>Host Operating Systems typically provide a Graphical User Interface that operates in pixel-mode and can be programatically sized by an application during its startup.</p> <p>Terminal emulators typically provide a Graphical-style User Interface that operates in character-mode but programatically sized by an application during its startup.</p> <p>Troubleshooting Hint:</p> <ul style="list-style-type: none"> Operating Systems typically provide user changeable properties to change the font and layout size of the shell

		<p>window in which the application will be run.</p> <ul style="list-style-type: none"> ▪ If the application aborts itself because it started in an undersized shell window, the operator should re-adjust the shell window's font and layout properties as appropriate.
2010/08/25	Non-compliant wxPython User Interface	<p>Mouse input is not always correct:</p> <ul style="list-style-type: none"> ▪ Mac OS X (Snow Leopard) - Only iTerm and WingIDE work; Terminal does not work. Incorrect characters under mouse reported for title of "widgets communicate". ▪ Cygwin (Windows XP) - Only Cygwin-X (xterm) works; console does not work.
2010/08/25	Partially-compliant wxPython User Interface	<p>Keyboard input subject to restrictions:</p> <ul style="list-style-type: none"> ▪ Apple "Cmd" (Command) Not supported on computer platforms running Mac OS X, Linux or Windows. ▪ Only "character" events supported; "key" press/release events Not supported. ▪ Only "Shift", "Ctrl" or "Alt" key can be pressed in combination with a character key; combinations of "Shift", "Ctrl" and "Alt" keys are Not supported. (Note: The "Alt" key event internally involves two separate "nCurses" getchar operations.)
2012/08/11	Partially-compliant wxPython User Interface	<p>Keyboard input subject to restrictions:</p> <ul style="list-style-type: none"> ▪ tsWxPasswordEntryDialog is derived from tsWxTextEntryDialog. The latter uses tsWxTextEditBox which has usability issues in both non-event driven and event driven modes. ▪ The non-event-driven mode uses curses.textpad.Textbox which monopolizes GUI platform and ignores mouse button clicks until the operator terminates input via double entry of control-G key. It also, responds to operator input after the control-G. ▪ The event driven mode uses an python emulation of curses.textpad.Textbox that provides enhanced edit capabilities while recognizing mouse button clicks and keyboard press-release activity. However, the display blinks as each new text character is displayed. More importantly, there is no mechanism to block the application program pending the operator's completion of text input.
2013/07/18	Partially-compliant wxPython User Interface	<ul style="list-style-type: none"> ▪ test tsWxGridSizer event processing generates too many event messages per button click. Button click event handling needs to be directly coupled to keyboard button window rather than coupled to the parent Frame provided handler.

7 NEW FEATURES

7.1 User Interface Displays:

- 1** Standard "wxPython" color names support "xterm-88color" and "xterm-256color" compatible displays.
- 2** Mapped substitution of standard "curses" 8-/16-color names for references to standard "wxPython" color names on "cygwin", "linux", "xterm", "xterm-color" and "xterm-16color" compatible displays.
- 3** Default foreground/background color support for "vt100" and "vt220" compatible black with single-color displays.
- 4** Built-in detection, creation and highlighting of CheckBox, RadioButton, MenuBar and TaskBar accelerator entries suitable for those platforms lacking a mouse, trackball or other pointing device. (Future)

7.2 High-Level Widget API

NOTES:

- 1) As a character-mode GUI-style toolkit, this product does not support those "wxPython" features associated with graphical elements such as bit images, icons, proportional-spaced fonts or HTML and XML text markup. The current release supports the porting of "wxPython" 2.8.9.2 application(s) to platforms running Python 2.5.x, 2.6.x, 2.7.x, 3.1.x and 3.2.x.
 - 2) Technical and resource issues drive the development effort. Initially, development focuses on establishing the feasibility of emulating core components of the the "wxPython" and associated "wxWidgets" API. Development then iteratively seeks to establish usability-enhancing components and to then to identify and resolve any appearance, behavior and API-conformance issues.
 - 3) See the unpublished "**tsWxGTUI_PyVx**" **Vol. 7 - Design Notes** (it was created only for personal use) for the identification and an overview of those currently implemented class modules, ones currently under construction and ones for which development applicability and/or commitments are still TBD.
-

The High-Level Widget API identifies the basic wxPython-style Graphical User Interface features that the Software Engineer can include for input and output interactions with the System Operator. It includes such widgets as the following:

- 1 **Button** - GUI object that may contain text or character-mode icons. If a mouse button is clicked when the cursor is over the object, an associated function or method will be initiated. The function or method will send a signal to notify other GUI objects of the event. Buttons are clickable windows that trigger an associated event (such as start, pause, resume or terminate an operation).
- 2 **Carat** - GUI object that may contain a set of special character-mode symbols, usually including a solid rectangle or a blinking underline character, showing the position where the typed text will appear.
- 3 **Check Box** - GUI object that may contain text or character-mode icons. If a mouse button is clicked when the cursor is over the object, it initiates an associated function or method. The function or method will toggle the check box to the next "ON", "OFF" or "TRI-STATE" value. The function or method will send a signal to notify other GUI objects of the event. Checkboxes are buttons to toggle the enabled or disabled state of an associated feature (such as start or stop logging)
- 4 **Cursor** - A special character-mode symbol, usually a solid rectangle or a blinking underline character, that signifies where the next character will be displayed on the screen.
- 5 **Dialog** - A GUI object with a title bar and sometimes a system menu, which can be moved around the screen. It can contain controls and other GUI objects and is often used to allow the user to make some choice or to answer a question. Dialogs are pop-up windows associated with an application task's menu bar (such as an input field for an operator's search criteria and an output field for a list of candidate WEB sites) that will be terminated upon completion.
- 6 **Frame** - GUI object whose size and position can (usually) be changed by the user. It usually has thick borders and a title bar, and can optionally contain a menu bar, toolbar and status bar. A frame can contain any GUI object that is not a frame or dialog. Frames are windows associated with an application task (such as an internet WEB Browser) that can be minimized into an icon, expanded to full screen or terminated upon operator demand.

- 7 Gauge** - GUI object whose horizontal or vertical bar shows a quantity (often time). It supports two working modes: determinate and indeterminate progress. First is the usual working mode while the second can be used when the program is doing some processing but you don't know how much progress is being done. In this case, you can periodically call the Pulse function to make the progress bar switch to indeterminate mode (graphically it's usually a set of blocks which move or bounce in the bar control). Gauges are used to indicate progress of an associated operation (scale with range such as 0% to %100%).
- 8 Menu** - GUI object that features a popup (or pull down) list of items, one of which may be selected before the menu goes away (clicking elsewhere dismisses the menu). It may be used to construct either menu bars or popup menus.
- 9 Menu Bar** - GUI object that features a series of menus accessible from the top of a frame. Each operator selectable entry triggers an associated event (such as create a file).
- 10 Panel** - GUI object that features a window on which controls are placed. Panels are usually placed within a frame. Its main feature over its parent window class is code for handling child windows and TAB traversal. Panels are container windows for other Lower Level GUI Objects.
- 11 Radio Box** - GUI objects that are used to select one of number of mutually exclusive choices. A radio box is displayed as a vertical column or horizontal row of labeled radio buttons. Radio Boxes are a collection of Buttons associated with the same group (such a AM or FM band) that are interdependent such that any one can become activated after the others simultaneously become deactivated.
- 12 Radio Button** - GUI object that may contain text or character-mode icons. If a mouse button is clicked when the cursor is over the object, it initiates an associated function or method. The function or method will turn the associated radio button "ON" and turn "OFF" all other radio buttons within the associated radio box. The function or method will send a signal to notify other objects of the event. Radio Buttons are used to activate one of the associated features (such as broadcast channel selection) after the other features associated with the same Radio Box group (such a AM or FM band) have become deactivated.
- 13 ScrollBar** - GUI object that includes a horizontal or vertical ScrollBarGauge between two ScrollBarButtons. The operator uses it to re-position (pan) the contents of an associated ScrolledText window.
- 14 ScrollBarButton** - GUI object that includes one arrow symbol ("<", ">", "^" or "V") that indicates the horizontal or vertical direction of scrolling. When the operator clicks on a ScrollBarButton, the contents of the associated ScrolledText window moves.
 - A single Left Click on one of the two horizontal ScrollBarButtons moves the text by one column. A rapid double Left Click on one of the two horizontal ScrollBarButtons moves the ScrolledText by one page (the horizontal column width). A single Right Click on one of the two horizontal ScrollBarButtons moves the text to the appropriate horizontally end point (either left/right most column).
 - A single Left Click on one of the two vertical ScrollBarButtons moves the text by one row. A rapid double Left Click on one of the two vertical ScrollBarButtons moves the ScrolledText by one page (the vertical row height). A single Right Click on one of the two vertical ScrollBarButtons moves the text to the appropriate vertical end point (top/bottom most row).
- 15 ScrollBarGauge** - GUI object located between two ScrollBarButtons, contains a bar graph that indicates the position and size of the displayed text relative to the non-displayed text.
 - When the bar graph is empty (blank), there is no text available to be displayed.

- When it is completely filled, all of the available text is displayed.
- When it is partially filled, the starting point of the graph displays the starting point of the displayed text relative to the available text. The size of the filled graph displays the size of the displayed text relative to the available text.
- When the operator single Left Clicks on a ScrollBarGauge between its two associated ScrollBarButtons, the contents of the ScrolledText window moves in proportion to the difference between the click and the associated text end positions.

- 16 Scrolled** - An application-independent GUI object base class for ScrolledWindow. It lays out the position and size of one ScrolledText window and one or two ScrollBars each with their associated ScrollBarGauge and pair of ScrollBarButtons. It enables an operator to select the columns and or rows of text to be fit and displayed within the available peep hole viewing area.
- 17 ScrolledText** - GUI object that allows one or more lines of text to be appended to a retained list. In conjunction with one or two associated pairs of ScrollBarButtons, it enables an operator to select the columns and or rows of text to be fit and displayed within the available peep hole viewing area.
- 18 ScrolledWindow** - GUI object that instantiates an application-specific instance of Scrolled to lay out the position and size of one ScrolledText window and one or two ScrollBars each with their associated ScrollBarGauge and pair of ScrollBarButtons. It enables an operator to select the columns and or rows of text to be fit and displayed within the available peep hole viewing area.
- 19 Splash Screen** - GUI object that simulates a pixel-mode bitmap image which is displayed upon application startup. It may contain text or character-mode icons. It features a window with a thin border and text describing the application. The bitmap-like display may be created from Panel, BoxSizer, GridSizer and TextCtrl widgets. It is created and shown during application initialization, before the application's own window. It either explicitly destroys itself or disappears after a it time-out.
- 20 Status Bar** - GUI object that feature a narrow window that can be placed along the bottom of a frame to give small amounts of status information. The object can contain one or more fields, one or more of which can be variable length according to the size of the window.
- 21 Static Text** - GUI object that features a text control that displays one or more lines of read-only text.
- 22 Text Ctrl** - GUI object that features a text control which allows text to be displayed and edited. It may be single line or multi-line. The text may be indented, line-wrapped and scrolled to fit the available display area.
- 23 Tool Bar (Future)** - GUI object that features a series of tool entries accessible from the top of a frame. Each operator selectable entry triggers an associated event that starts the selected tool in a new Frame.

NOTES:

- 1) Event Handling support currently associates mouse clicks with the enclosing GUI object that is not obscured by overlaying GUI objects. The toolkit generates an event notification and dispatches it to the event handler designated by the application. It will dispatch unhandled event notifications to a default handler.
 - 2) Keyboard Input Events are detected. The raw characters are echoed but are NOT currently forwarded to the window object having focus.
 - 3) Mouse Input Events are detected. The x-y coordinates and button state are echoed. Single Left Clicks, Double Left Click and Right Clicks are forwarded to the window object positioned to receive and process the event. More complex GUI event detection, analysis and processing is not yet supported. Platform-specific vt100/vt220 terminal emulators do NOT conform to the xterm/xterm-color mouse click event protocol recognized by nCurses. Instead of a single mouse click event, the vt100/vt220 terminal emulators generate a string of escape sequence events. It is unknown if the "tsWxGTUI_PyVx" Toolkit can develop logic to synthesize an xterm/xterm-color mouse click event from the string of vt100/vt220 escape sequence events.
 - 4) Automatic layout support is currently provided by the BoxSizer and GridSizer widgets or by combinations of them. More complex GUI object positioning and sizing is not yet supported.
-

7.3 GUI Events

The "tsWxPyApp" module invokes the "tsWxEventLoop.tsGetGraphicalUserInput" method to wait for the following system events:

- TimerEvent - One second interval signal.
- MouseEvent - Left/Middle/Right mouse button and Wheel mouse movement signals.
- KeyEvent - Shift/Ctrl/Alt, alphabetic, numeric, punctuation, and other printable character keyboard button pressed/release signals.

The tsWxEventLoop.tsThisIsGuiObject" method identifies the "objectId" of the GUI object (frame, dialog, checkbox, radio box, button, textctrl etc.) to receive each mouse and key event.

7.4 Keyboard & Mouse Input

```
#-----  
#"Time-stamp: <07/01/2015  8:28:09 AM rsg>"  
#-----
```

```
==== Title Page for File: README9-KeyboardMouseInput.txt ===
```

```
+-----+-----+ TeamSTARS "tsWxGTUI_PyVx" Toolkit  
| ts | Wx |       with Python 2x & Python 3x based  
+-----+-----+       Command Line Interface (CLI)  
| G T U I |       and "Curses"-based "wxPython"-style,  
+-----+-----+       Graphical-Text User Interface (GUI)
```

Get that cross-platform, pixel-mode "wxPython" feeling on character-mode 8-/16-color (xterm-family) & non-color (vt100-family) terminals and terminal emulators.

You can find this and other plain-text files in the Toolkit subdirectory named:

```
"./<Toolkit Recipient's Repository>/Documents".
```

```
<Your Working Repository>  
(e.g. "tsWxGTUI_PyVx_Preview")
```

```
|  
| Working repository containing directories and  
| files to be packaged into downloadable "tarball"  
| and/or "zip" files via the setup shell scripts  
| at the bottom of this diagram.  
|
```

```
+-- ["Documents"]
```

```
|  
| | This directory contains a collection of files  
| | which provide the Toolkit recipient with an  
| | understanding of the purpose, goals & capabilities,  
| | non-goals & limitations, terms & conditions and procedures for installing, operating,  
| | modifying and redistributing the Toolkit.  
| |
```

```
| | Deliverable Toolkit manual pages are a  
| | form of online software documentation  
| | usually found on a Unix or Unix-like operating system.  
| |
```

```
| +-- "README9-KeyboardMouseInput.txt"
```

```
| +-- "README3-Documents.txt"
```

```
+-- ["ManPages"]
```

```
+-- ["Notebooks"]
```

```
+-- ["SourceDistributions"]  
|
```

```
+-- "README.txt"
```

```
===== TABLE OF CONTENTS =====
```

1. Software design and implementation constraints.
2. wxWidget Accelerator Hotkeys
3. wxWidget Catching key events globally
4. Keyboard Shortcut Keys
5. Table of keyboard shortcuts
6. Computer keyboard shortcut keys
7. Microsoft Windows shortcut keys
8. Apple Macintosh shortcut keys
9. Unix and Linux shortcut keys
10. Top 10 keyboard shortcuts everyone should know
11. Mouse Position and Button Input

```
===== Software design and implementation constraints =====
```

1. Software design and implementation constraints

The xterm-style 8-/16-color terminals and terminal emulators accept operator input via keyboard and mouse devices.

The vt100-style non-color terminals and terminal emulators typically accept operator input only via keyboard devices. A few generate non-xterm-style mouse events which precipitate application traps.

Operator input must trigger an event notification that must be sent to and then handled by the target GUI object such as a button or menu list item.

```
*****
**          FUTURE DESIGN AND IMPLEMENTATION CHANGE          **
*****
```

Consequently, in order to support color and non-color terminals and terminal emulators, each button and menu list item must provide and use an accelerator hotkey as well as an accelerator hotkey by which the operator may change focus of a top-level GUI object (frame or dialog) from background to foreground thereby relegating the previous foreground object to the background.

Event handlers for vt100-style non-color terminals and terminal emulators must either disable or handle mouse input event notifications.

NOTE:

The implementation of keyboard shortcuts has been and should continue to be postponed until after the resolution of GUI object focus changes using the curses panel capability. It being simpler to verify focus changes triggered via a mouse than triggered via keyboard input and recursive table lookup.

===== wxWidget Accelerator Hotkeys =====

2. wxWidget Accelerator Hotkeys

Excerpt From

https://books.google.com/books?id=CyMsvtgng0QC&pg=PA180&lpg=PA180&dq=wxWidget+Accelerator+Hotkeys&source=bl&ots=SU7wm9DnDc&sig=9_ioDyoPjZJNF2TQhQVG_6OtJLM&hl=en&sa=X&ei=NfBzVbn2JMrVsAWH6oDYBg&ved=0CD0Q6AEwBA#v=onepage&q=wxWidget%20Accelerator%20Hotkeys&f=false:

"Accelerators

An accelerator implements a keyboard shortcut for a menu command, enabling the user to execute the command quickly. These shortcuts take precedence over other keyboard processing, such as EVT_CHAR handlers. Standard shortcuts include Ctrl-O to open a file and Ctrl-V to paste data into the application. The easiest way to implement accelerators is to specify them in menu items. For example:

```
menu->Append(wxID_COPY, wxT("Copy\tCtrl+C"));
```

wxWidgets interprets the text after the "tab" character as an accelerator and adds it to the menu's accelerator table. In this example, when the user presses Ctrl-C the wxID_COPY command is sent, just as though the menu item was selected...."

===== wxWidget Catching key events globally =====

3. wxWidget Catching key events globally

Excerpt From https://wiki.wxwidgets.org/Catching_key_events_globally:

"Keyboard events go to the component that currently has focus and do not propagate to the parent; if you are trying to catch key events globally it can thus be a little tricky. Here are a few ways to solve this problem - keep in mind there are probably more than presented here.

Before getting started, some helpful notes about cases where you may be catching the wrong event or where you may not need global key catching at all:

- * Many components will only receive key events if they have the `wxWANTS_CHARS` style flag enabled; then, you need to catch `EVT_CHAR` rather than or in addition to `EVT_KEY_DOWN`.
- * For catching Enter presses on text controls, use style flag `wxTE_PROCESS_ENTER`, and catch event `EVT_TEXT_ENTER`."

===== Keyboard Shortcut Keys =====

4. Keyboard Shortcut Keys

Excerpts From Wikipedia, the free encyclopedia:

" For Wikipedia keyboard shortcuts,
see Wikipedia:Keyboard shortcuts.

For a list of keyboard shortcuts,
see Table of keyboard shortcuts.

In computing, a keyboard shortcut is a series of one or several keys that invoke a software or operating system operation (in other words, cause an event) when triggered by the user. The meaning of term "keyboard shortcut" can vary depending on software manufacturer. For instance, Microsoft differentiates keyboard shortcuts from hotkeys ("mnemonics" on Windows) whereby the former consists of a specific key combination used to trigger an action, and the latter represents a designated letter in a menu command or toolbar button that when pressed together with the Alt key, activates such command --- whereas a "hotkey" on Windows is a system wide shortcut that is always available in all contexts as long as the program responsible for it is running and not suspended.

Description

Keyboard shortcuts are typically a means for invoking one or more commands using the keyboard that would otherwise be accessible only through a menu, a pointing device, different levels of a user interface, or via a command-line interface. Keyboard shortcuts are generally used to expedite common operations by reducing input sequences to a few keystrokes, hence the term "shortcut".[1]

To differentiate from general keyboard input, most keyboard shortcuts require the user to press and hold several keys simultaneously or a sequence of keys one after the other. Unmodified key presses are sometimes accepted when the keyboard is not used for general input - such as with graphics packages e.g. Adobe Photoshop or IBM Lotus Freelance Graphics. Other keyboard shortcuts use function keys that are dedicated for use in shortcuts and may only require

a single keypress. For simultaneous keyboard shortcuts, one usually first holds down the modifier key(s), then quickly presses and releases the regular (non-modifier) key, and finally releases the modifier key(s). This distinction is important, as trying to press all the keys simultaneously will frequently either miss some of the modifier keys, or cause unwanted auto-repeat. Sequential shortcuts usually involve pressing and releasing a dedicated prefix key, such as the Esc key, followed by one or more keystrokes.

Mnemonics are distinguishable from keyboard shortcuts. One difference between them is that the keyboard shortcuts are not localized on multi-language software but the mnemonics are generally localized to reflect the symbols and letters used in the specific locale. In most GUIs, a program's keyboard shortcuts are discoverable by browsing the program's menus --- the shortcut is indicated next to the menu choice. There are keyboards that have the shortcuts for a particular application already marked on them. These keyboards are often used for editing video, audio, or graphics,[2] as well as in software training courses. There are also stickers with shortcuts printed on them that can be applied to a regular keyboard. Reference cards intended to be propped up in the user's workspace also exist for many applications. In the past, when computer hardware was more standardized, it was common for computer books and magazines to print cards that were cut out, intended to be placed over the user's keyboard with the printed shortcuts noted next to the appropriate keys...."

===== Table of keyboard shortcuts =====

5. Table of keyboard shortcuts

From Wikipedia, the free encyclopedia:

Excerpt From "http://en.wikipedia.org/wiki/Table_of_keyboard_shortcuts":

===== Computer keyboard shortcut keys =====

6. Computer keyboard shortcut keys

Excerpt From "<http://www.computerhope.com/shortcut.htm>":

===== Microsoft Windows shortcut keys =====

7. Microsoft Windows shortcut keys

Excerpt From "<http://www.computerhope.com/shortcut/windows.htm>":

===== Apple Macintosh shortcut keys =====

8. Apple Macintosh shortcut keys

Excerpt From "Apple Macintosh shortcut keys":

===== Unix and Linux shortcut keys =====

9. Unix and Linux shortcut keys

Excerpt From "<http://www.computerhope.com/ushort.htm>":

===== Top 10 keyboard shortcuts everyone should know =====

10. Top 10 keyboard shortcuts everyone should know

Excerpt From "<http://www.computerhope.com/tips/tip79.htm>":

"In computing, source code is any collection of computer instructions (possibly with comments) written using some human-readable computer language, usually as text. The source code of a program is specially designed to facilitate the work of computer programmers, who specify the actions to be performed by a computer mostly by writing source code. The source code is often transformed by a compiler program into low-level machine code understood by the computer. The machine code might then be stored for execution at a later time. Alternatively, an interpreter can be used to analyze and perform the outcomes of the source code program directly on the fly.

Most computer applications are distributed in a form that includes executable files, but not their source code. If the source code were included, it would be useful to a user, programmer, or system administrator, who may wish to modify the program or to understand how it works.

Aside from its machine-readable forms, source code also appears in books and other media; often in the form of small code snippets, but occasionally complete code bases; a well-known case is the source code of PGP."

===== Mouse Position and Button Input =====

11. Mouse Position and Button Input

Excerpts From Wikipedia, the free encyclopedia:

"In computing, a mouse is a pointing device that detects two-dimensional motion relative to a surface. This motion is typically translated into the motion of a pointer on a display, which allows for fine control of a graphical user interface.

Picture:

"A computer mouse with the most common standard features: two buttons and a scroll wheel, which

can also act as a third button."

Physically, a mouse consists of an object held in one's hand, with one or more buttons. Mice often also feature other elements, such as touch surfaces and "wheels", which enable additional control and dimensional input."

11.1 Operation with "modern" 8/16 color xterm-family terminal emulators

The mouse (trackball, touch screen, touch pad) is moved until a cursor on the display is directly above the graphical user interface object (button, check box, radio button, scrollbar gauge or slider which can trigger a response to the mouse input. Press and release one of the mouse buttons one or more times in quick succession (click, double-click or tripple-click). Mouse interactions mimic those typical of Linux, Mac OS X, Microsoft Windows and Unix GUI interfaces.

11.1.1 The Python Curses module reports the following mouse input data:

- a. `mouseId` --- First, Second, Third
- b. `x` --- Display horizontal column position (0=Left)
- c. `y` --- Display vertical row position (0=Top)
- d. `z` --- Reserved for display depth position
- e. `bstate` --- Mouse button state (button ID, single-/double-triple-clicked, pressed/released)

11.1.2 The mouse button is typically the primary button for normal operations.

A few examples:

- a. A left click on a scroll bar arrow button selects the function associated with the button (scroll one column to the left or right, scroll one row up or down).
- b. A left double-click on a scroll bar arrow button selects the function associated with the button (scroll one screen page width to the left or right, scroll one page screen height up or down).
- c. A right click on a scroll bar arrow selects the function associated with the button (scroll to the left or right most text column, scroll to

the top or bottom most text row.

- d. A left click on a scroll bar gauge (the area that displays the relative amount and position of the text being displayed) moves the selected text into the display area.

11.2 Operation with "ancient" non-color vt100-family terminal emulators

The mouse (trackball, touch screen, touch pad) is moved until a cursor on the display is directly above the graphical user interface object (button, check box, radio button, scrollbar gauge or slider which can trigger a response to the mouse input. Press and release one of the mouse buttons (click). Mouse interactions mimic those typical of Linux, Mac OS X, Microsoft Windows and Unix GUI interfaces.

11.2.1 The Python Curses module reports the following mouse input data via one (when pressed or released) or more (when pressed and quickly released) escape prefixed six-character strings (the following was deduced from data capture and print out):

- a. escape character --- (27 = 0x1b)
- b. device id1 --- (91 = unknown)
- c. device id2 --- (77 = unknown)
- d. button + state + 32 --- (0 = left button;
1 = middle button;
2 = right button;
4 = wheel mouse;
5 = wheel mouse;
32 = pressed state;
35 = released state)
- e. x + 33 --- Display horizontal column position
(0=Left)
- f. y + 33 --- Display vertical row position
(0=Top)

11.2.2 The mouse button is typically the primary button for normal operations.

A few examples:

- a. A left click on a scroll bar arrow button selects the function associated with the button (scroll one column to the left or right, scroll one row up or down).

- b. A left click on a scroll bar gauge (the area that displays the relative amount and position of the text being displayed) moves the selected text into the display area.

===== End-Of-File =====

Draft

8 APPLICATION NOTES

8.1 README-GettingStarted

Draft

```
#-----
#"Time-stamp: <06/24/2015  2:46:38 PM rsg>"
#-----
```

```
===== Title Page for File: GETTING_STARTED.txt =====
```

```
+-----+-----+ TeamSTARS "tsWxGTUI_PyVx" Toolkit
| ts | Wx |          with Python 2x & Python 3x based
+-----+-----+          Command Line Interface (CLI)
| G T U I |          and "Curses"-based "wxPython"-style,
+-----+-----+          Graphical-Text User Interface (GUI)
```

Get that cross-platform, pixel-mode "wxPython" feeling on character-mode 8-/16-color (xterm-family) & non-color (vt100-family) terminals and terminal emulators.

You can find this and other plain-text files in the Toolkit subdirectory named:

"./<Toolkit Recipient's Repository>/Documents".

<Your Working Repository>

(e.g. "tsWxGTUI_PyVx_Preview-Site-Packages")

```
|
| Working repository containing directories and
| files to be packaged into downloadable "tarball"
| and/or "zip" files via the setup shell scripts
| at the bottom of this diagram.
|
```

+-- ["Documents"] (Original)

```
|
| | This directory contains a collection of files
| | which provide the Toolkit recipient with an
| | understanding of the purpose, goals & capabilities,
| | non-goals & limitations, terms & conditions and
| | procedures for installing, operating, modifying and
| | redistributing the Toolkit.
| |
```

+-- "GETTING_STARTED.txt"

+-- ["ManPages"] (Original)

```
|
| | Deliverable Toolkit manual pages are a
| | form of online software documentation
| | usually found on a Unix or Unix-like operating
| | system.
| |
```

```
|
| | Topics covered include computer programs
| | (including library and system calls),
| | formal standards and conventions, and even
| | abstract concepts.
| |
```

```
|
| | A user may NOT invoke a man page by issuing the
| | man command. Instead, a user may display a man
| | page by issuing the less <man document file>
| | command.
| |
```

```

|      +-- ["tsManPagesLibCLI"]
|      +-- ["tsManPagesLibGUI"]
|      +-- ["tsManPagesTestsLibCLI"]
|      +-- ["tsManPagesTestsLibGUI"]
|      +-- ["tsManPagesToolsCLI"]
|      +-- ["tsManPagesToolsGUI"] (Future)
|      +-- ["tsManPagesToolsLibCLI"]
|      +-- ["tsManPagesToolsLibGUI"] (Future)
|      +-- ["tsManPagesUtilitiesCLI"] (Future)
|
+-- ["Notebooks"] (Pre-dates Documents)
|
|   Contains a collection of commentaries that
|   express opinions or offerings of explana-
|   tions about events or situations that might
|   be useful to Toolkit installers, developers,
|   operators, troubleshooters and distributors.
|   The documents may be in Application-specific
|   formats (Adobe PDF, JPEG Bit-mapped image,
|   Microsoft Office, Plain text etc.).
|
|   The collection includes:
|
|   a) DeveloperDocuments (API-References-
|       Pixel-Mode-wxPython and Developer-
|       ReadMe-Files); and
|
|   b) EngineeringDocuments (Product Marketing
|       Documentation, Project Documentation and
|       Technical Documentation). (Pre-dates Documents)
|
+-- ["SourceDistributions"] (Original)
|
|   Contains a collection of computer program
|   source code files that the Toolkit recip-
|   ient will need to install, operate, modify
|   and re-distribute the Toolkit.
|
+-- ["Developer-Sandboxes"] (Pre-dates Site-Packages)
|
|   A sandbox is a testing environment that iso-
|   lates untested code changes and outright
|   experimentation from the production environ-
|   ment or repository.
|
+-- ["tsWxGTUI_PyVx"] (Developer-Sandbox)
|
|   +-- ["Documents"] (Copy)
|
|   This directory contains a collection of files
|   which provide the Toolkit recipient with an
|   understanding of the purpose, goals & capabil-
|   ities, non-goals & limitations, terms & condi-
|   tions and procedures for installing, operating,
|   modifying and redistributing the Toolkit.

```

```

+--- ["ManPages"] (Copy)
|
| Deliverable Toolkit manual pages are a
| form of online software documentation
| usually found on a Unix or Unix-like oper-
| ating system.
|
| Topics covered include computer programs
| (including library and system calls),
| formal standards and conventions, and even
| abstract concepts.
|
| A user may NOT invoke a man page by iss-
| uing the man command. Instead, a user may
| display a man page by issuing the
| less <man document file> command.
|
+--- ["tsManPagesLibCLI"]
+--- ["tsManPagesLibGUI"]
+--- ["tsManPagesTestsLibCLI"]
+--- ["tsManPagesTestsLibGUI"]
+--- ["tsManPagesToolsCLI"]
+--- ["tsManPagesToolsGUI"] (Future)
+--- ["tsManPagesToolsLibCLI"]
+--- ["tsManPagesToolsLibGUI"] (Future)
+--- ["tsManPagesUtilitiesCLI"] (Future)
+--- ["Python-2x"] (Developer-Sandbox)
|
| +--- ["tsWxGTUI_Py2x"]
+--- ["Python-3x"] (Developer-Sandbox,
| Ported from Python-2x)
|
+--- ["tsWxGTUI_Py3x"]
+--- ["Site-Packages"]
|
| Site-packages is the location where third-
| party packages are installed (i.e., those
| not part of the core Python distribution).
| NOTE: That with Linux, Mac OS X and Unix
| operating systems one must have root priv-
| ileages to write to that location.
|
+--- ["tsWxGTUI_PyVx"] (Site-Package)
|
+--- ["Documents"] (Copy)
|
| This directory contains a collection of files
| which provide the Toolkit recipient with an
| understanding of the purpose, goals & capabil-
| ities, non-goals & limitations, terms & condi-
| tions and procedures for installing, operating,
| modifying and redistributing the Toolkit.

```



```

|                                     +-- ["ManPages"] (Copy)
|                                     |
|                                     | Deliverable Toolkit manual pages are a
|                                     | form of online software documentation
|                                     | usually found on a Unix or Unix-like oper-
|                                     | ating system.
|                                     |
|                                     | Topics covered include computer programs
|                                     | (including library and system calls),
|                                     | formal standards and conventions, and even
|                                     | abstract concepts.
|                                     |
|                                     | A user may NOT invoke a man page by issu-
|                                     | ing the man command. Instead, a user may
|                                     | display a man page by issuing the
|                                     | less <man document file> command.
|                                     |
|                                     +-- ["tsManPagesLibCLI"]
|                                     +-- ["tsManPagesLibGUI"]
|                                     +-- ["tsManPagesTestsLibCLI"]
|                                     +-- ["tsManPagesTestsLibGUI"]
|                                     +-- ["tsManPagesToolsCLI"]
|                                     +-- ["tsManPagesToolsGUI"] (Future)
|                                     +-- ["tsManPagesToolsLibCLI"]
|                                     +-- ["tsManPagesToolsLibGUI"] (Future)
|                                     +-- ["tsManPagesUtilitiesCLI"] (Future)
|
|                                     +-- ["Python-2x"] (Site-Package)
|                                     |
|                                     | +-- ["tsWxGTUI_Py2x"]
|                                     |
|                                     +-- ["Python-3x"] (Site-Package,
|                                     | Ported from Python-2x)
|                                     |
|                                     | +-- ["tsWxGTUI_Py3x"]
|                                     |
|
+-- "MANIFEST.in"
|
| Deliverable File inclusion criteria list.
|
+-- "MANIFEST_template.in"
|
| Deliverable Generic file inclusion criteria list
| template for any Python version-specific TeamSTARS
| "tsWxGTUI_PyVx" Toolkit.
|
+-- "MANIFEST_TREE.html"
|
| Non-Deliverable Diagram (Multi-Level Org Chart)
| depicting the hierarchical relationship between files
| in the release, in Hypertext Markup Language format.
|
| Diagram created via Command "./MANIFEST_TREE.sh".
|
+-- "MANIFEST_TREE.sh"

```

```

|
| Deliverable POSIX-style Command Line Interface shell
| script to generate diagrams depicting the hierarchical
| relationship between files in the release
| ("MANIFEST_TREE.html" and "MANIFEST_TREE.txt").
|
+-- "MANIFEST_TREE.txt"
|
| Non-Deliverable Diagram (Multi-Level Org Chart)
| depicting the hierarchical relationship between
| files in the release, in Plain Text format.
|
| Diagram created via Command "./MANIFEST_TREE.sh".
|
+-- "setup_tsWxGTUI_PyVx_Preview_tar_file.sh"
|
| Deliverable POSIX-style Command Line Interface shell
| script to generate downloadable "tarball" file.
|
+-- "setup_tsWxGTUI_PyVx_Preview_zip_file.sh"
|
| Deliverable POSIX-style Command Line Interface shell
| script to generate downloadable "zip" file.
|
+-- "README2-Repository.txt"

```

===== TABLE OF CONTENTS =====

1. What is in the software distribution?
 - 1.1 What is the TeamSTARS "tsWxGTUI_PyVx" Toolkit?
 - 1.2 How to prepare your computer(s) for use with the Toolkit?
 - 1.3 How can you become familiar with the features, look and feel of the Toolkit?
 - 1.4 What are the currently known Toolkit limitations, bugs and update roadmap?
2. User Interfaces
 - 2.1 Command Line Interface
 - 2.2 Graphical User Interface
 - 2.2.1 C/C++ packages
 - 2.2.2 Packages in other languages
 - 2.3 Window Manager
3. Operating System
 - 3.1 Linux (POSIX-compatible CLI & GUI)
 - 3.2 Mac OS X (POSIX-compatible CLI & GUI)

3.3 Microsoft Windows

- 3.3.1 "Cygwin" (POSIX-compatible CLI & GUI)
- 3.3.2 "GnuWin32" (POSIX-compatible CLI only)
- 3.3.3 "Windows PowerShell" (Windows-native CLI only)
- 3.3.4 "Command Prompt" (Windows-native CLI only)

3.4 Unix (POSIX-compatible CLI & GUI)

4. Toolkit Development Resources

4.1 Python Programming Language Resources

- 4.1.1 Python 3.x
- 4.1.2 Python 2.x

4.2 Open Source & Commercial Python Resources

- 4.2.1 Python Software Foundation Website
- 4.2.2 ActiveState Website

4.3 Python Training Resources

- 4.3.1 Dive into Python
- 4.3.2 Learn Python
- 4.3.3 Python Community
- 4.3.4 Python Cookbook

5. Python Download Gotchas

- 5.1 Linux and Unix Operating Systems (CAUTION)
- 5.2 Microsoft Windows Operating Systems (WARNING)

6. wxPython/wxWidgets Development Resources

6.1 wxPython Programming Language Resources

- 6.1.1 wxPython 2.8.9.2 (Currently Emulated API)
- 6.1.2 wxPython 3.0.2.0 (Future Emulated API)

6.2 Open Source wxPython/wxWidgets Resources

- 6.2.1 wxPython.org Website
- 6.2.2 wxWidgets.org Website

6.3 wxPython/wxWidgets Training Resources

- 6.3.1 wxPython Community
- 6.3.2 wxPython Cookbook

===== README =====

1. What is in the software distribution?

Browse through the following information located in the directory `"/tsWxGTUI_PyVx/Documents"`. It provides an overview of the Toolkit distribution and its contents:

1.1 What is the TeamSTARS `"tsWxGTUI_PyVx"` Toolkit?

- a) `"README1-Introduction.txt"`
- b) `"README2-Repository.txt"`
- c) `"README3-Documents.txt"`
- d) `"README4-ManPages.txt"`
- e) `"README5-Notebooks.txt"`
- f) `"README6-SourceDistribution.txt"`
- g) `"README7-DeveloperSandboxes.txt"`
- h) `"README8-SitePackages.txt"`
- i) `"README9-KeyboardMouseInput.txt"`

1.2 How to prepare your computer(s) for use with the Toolkit?

- a) `"GETTING_STARTED.txt"`

1.3 How can you become familiar with the features, look and feel of the Toolkit?

- a) `"DEMO.txt"`
- b) `"TROUBLESHOOT.txt"`

1.4 What are the currently known Toolkit limitations, bugs and update roadmap?

- a) `"BUGS.txt"`
- b) `"TO-DO.txt"`

===== USER INTERFACES =====

2. User Interfaces

2.1 Command Line Interface

From Wikipedia, the free encyclopedia:

"A command-line interface or command language interpreter (CLI), also known as command-line user interface, console user interface,[1] and character user interface (CUI), is a means of interacting with a computer program where the user (or client) issues commands to the program in the form of successive lines of text (command lines).

The CLI was the primary means of interaction with most computer systems until the introduction of the video display terminal in the mid-1960s, and continued to be used throughout the 1970s and 1980s on OpenVMS, Unix systems and personal computer systems including MS-DOS, CP/M and Apple DOS. The interface

is usually implemented with a command line shell, which is a program that accepts commands as text input and converts commands to appropriate operating system functions.

Command-line interfaces to computer operating systems are less widely used by casual computer users, who favor graphical user interfaces. Command-line interfaces are often preferred by more advanced computer users, as they often provide a more concise and powerful means to control a program or operating system.

Programs with command-line interfaces are generally easier to automate via scripting.

Alternatives to the command line include, but are not limited to text user interface menus (see IBM AIX SMIT for example), keyboard shortcuts, and various other desktop metaphors centered on the pointer (usually controlled with a mouse)."

2.2 Graphical User Interface

From Wikipedia, the free encyclopedia:

"In computing, a graphical user interface (GUI,[1] sometimes pronounced "gooey" or "jee-you-eye") [2] is a type of interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, as opposed to text-based interfaces, typed command labels or text navigation. GUIs were introduced in reaction to the perceived steep learning curve of command-line interfaces (CLIs), [3] [4] [4] which require commands to be typed on the keyboard.

The actions in a GUI are usually performed through direct manipulation of the graphical elements.[5] In addition to computers, GUIs can be found in hand-held devices such as MP3 players, portable media players, gaming devices and smaller household, office and industry equipment. The term "GUI" tends not to be applied to other low-resolution types of interfaces with display resolutions, such as video games (where HUD[6] is preferred), or not restricted to flat screens, like volumetric displays[7] because the term is restricted to the scope of two-dimensional display screens able to describe generic information, in the tradition of the computer science research at the PARC (Palo Alto Research Center)."

From Wikipedia, the free encyclopedia:

"This is a list of packages implementing a platform-independent GUI (PIGUI). These can be used to develop

software that can be ported to multiple platforms without changes to its source code.

2.2.1 C/C++ packages

Excerpts From Wikipedia, the free encyclopedia:

Name	Owner	Platforms	License
GTK+	GNOME Foundation	X11, Windows, DirectFB, Quartz on Mac OS X	LGPL
MKS Toolkit	DataFocus, Inc.	Windows from X11 code	Commer- cial
Qt	Qt Project	Windows, Linux(X11), OS-X, iOS, Android [1]	LGPL, GPL, Commer- cial
wxWidgets	wxWidgets team	Windows, OS/2, X11, Mac OS X, iOS	LGPL

2.2.2 Packages in other languages

Excerpts From Wikipedia, the free encyclopedia:

Name	Owner	Platforms	License
Java/Swing	Oracle/ Sun Micro- systems	Windows, OS/2, X11, Mac OS X	Free
Tcl/Tk	Open Source }	Windows, OS/2, X11, Mac OS X	Free

2.3 Window Manager

From Wikipedia, the free encyclopedia:

"A window manager is system software that controls the placement and appearance of windows within a windowing system in a graphical user interface.[1]

Most window managers are designed to help provide a desktop environment. They work in conjunction with the underlying graphical system that provides required functionality---support for graphics hardware, pointing devices, and a keyboard, and are often written and created using a widget toolkit.

Few window managers are designed with a clear distinction between the windowing system and the window manager. Every graphical user interface based on a windows metaphor has some form of window management. In practice, the elements of this functionality vary greatly.[2] Elements usually associated with window managers allow the user to open, close, minimize, maximize, move, resize, and keep track of running windows, including window decorators. Many window managers also come with various utilities and features: e.g. docks, task bars, program launchers, desktop icons, and wallpaper."

===== OPERATING SYSTEM =====

3. Operating System

The TeamSTARS "tsWxGTUI_PyVx" Toolkit software source code is designed to be used with four popular operating system classes that are significantly different.

Use of the designated Python programming language release(s) and associated operating system specific implementations of the Python Virtual Machine and Interpreter should make it unnecessary to ever modify any of the Toolkit source code to accomodate different operating systems or computer hardware. Of course, you may modify the Toolkit source code if you cannot wait for someone else to port it to a newer or even older Python programming language release.

The following third-party reference material has been selected to ensure that Toolkit recipients, users and authors share the same vision of system capabilities and limitations. Providing you only with links to the reference material would not be as convenient for you. The third-party material might unexpectedly be deleted, moved or lose its relevance by being radically changed.

From Wikipedia, the free encyclopedia

"An operating system (OS) is software that manages computer hardware and software resources and provides common services for computer programs. The operating system is an essential component of the system software in a computer system. Application programs usually require an operating system to function.

Time-sharing operating systems schedule tasks for efficient use of the system and may also include accounting software for cost allocation of processor time, mass storage, printing, and other resources.

For hardware functions such as input and output and memory allocation, the operating system acts as an intermediary between programs and the computer hardware,[1][2] although the application code is usually executed directly by the hardware and will frequently make a system call to an OS function or be interrupted by it. Operating systems can be found on almost any device that contains a computer -- from cellular phones and video game consoles to supercomputers and web servers.

Examples of popular modern operating systems include Android, BSD, iOS, Linux, OS X, QNX, Microsoft Windows,[3] Windows Phone, and IBM z/OS. All these examples, except Windows, Windows Phone and z/OS, share roots in UNIX."

Excerpts From Wikipedia, the free encyclopedia

"POSIX (/poziks/ poz-iks), an acronym for Portable Operating System Interface,[1] is a family of standards specified by the IEEE Computer Society for maintaining compatibility between operating systems. POSIX defines the application programming interface (API), along with command line shells and utility interfaces, for software compatibility with variants of Unix and other operating systems.[2][3]"

"Name

Originally, the name "POSIX" referred to IEEE Std 1003.1-1988, released in 1988. The family of POSIX standards is formally designated as IEEE 1003 and the international standard name is ISO/IEC 9945.

The standards emerged from a project that began circa 1985. Richard Stallman suggested the name POSIX to the IEEE instead of former IEEE-IX. The committee found it more easily pronounceable and memorable, and thus adopted it.[2][4]

Overview

The POSIX specifications for Unix-like operating systems originally consisted of a single document for the core programming interface, but eventually grew to 19 separate documents (POSIX.1, POSIX.2, etc.).[5] The standardized user command line and scripting interface were based on the Korn shell. Many user-level programs, services, and utilities including awk, echo, ed were also standardized, along with required program-level services including basic I/O (file, terminal, and network) services. POSIX also defines a standard threading library API which is supported by most modern operating systems. Nowadays,

most of POSIX parts are combined into a single standard, IEEE Std 1003.1-2008, also known as POSIX.1-2008.

As of 2014, POSIX documentation is divided in two parts:

- * POSIX.1, 2013 Edition: POSIX Base Definitions, System Interfaces, and Commands and Utilities (which include POSIX.1, extensions for POSIX.1, Real-time Services, Threads Interface, Real-time Extensions, Security Interface, Network File Access and Network Process-to-Process Communications, User Portability Extensions, Corrections and Extensions, Protection and Control Utilities and Batch System Utilities. This is POSIX 1003.1-2008 with Technical Corrigendum 1.)
- * POSIX Conformance Testing: A test suite for POSIX accompanies the standard: VSX-PCTS or the VSX POSIX Conformance Test Suite.[6]

The development of the POSIX standard takes place in the Austin Group, a joint working group linking the IEEE, The Open Group and the ISO/IEC JTC 1 organizations."

3.1 Linux (POSIX-compatible CLI & GUI)

From Wikipedia, the free encyclopedia

"Linux is a Unix-like and mostly POSIX-compliant computer operating system.

The defining component of Linux is the Linux kernel, a computer program (first released on 5 October 1991 by Linus Torvalds) that manages input/output requests from software, and translates them into data processing instructions for the central processing unit and other electronic components of a computer.

The Free Software Foundation uses the name GNU/Linux to describe the operating system as a whole because its GNU toolchain software is orders of magnitude larger and supplies the Unix-like Application Programming Interface (API) and functionality.

Linux was originally developed as a free operating system for Intel x86-based personal computers, but has since been ported to more computer hardware platforms than any other operating system. It is the leading operating system on servers, main-frame computers and supercomputers, but is used on only around 1% of desktop computers.

Linux also runs on embedded systems, which are devices whose operating system is typically built into the firmware and is highly tailored to the system; this includes mobile phones, tablet computers, network

routers, facility automation controls, televisions and video game consoles.

For a description comparing the popular, inexperienced user-friendly Ubuntu (whose early motto was "Linux for human beings") with its Debian foundation (whose motto is "the universal operating system") you should read the following:

["https://www.wikivs.com/wiki/Debian_vs_Ubuntu"](https://www.wikivs.com/wiki/Debian_vs_Ubuntu)

For a well written introduction to embedded Linux, you should read the following blog:

["http://www.embedded.com/electronics-blogs/open-mike/4420567/Learning-Linux-for-embedded-systems"](http://www.embedded.com/electronics-blogs/open-mike/4420567/Learning-Linux-for-embedded-systems)

Android, the most widely used operating system for tablets and smartphones, is built on top of the Linux kernel."

3.1.1 Character-mode Command Line Interface

The Linux "Terminal" application and "bash" shell support the TeamSTARS "tsWxGTUI_PyVx" Toolkit with its local Python-based Command Line Interface (CLI).

The "bash" shell, with its "ssh" and "sftp" commands, enables the operator to connect with remote computers and thereby concurrently conduct not only one or more local CLI sessions but also one or more remote CLI sessions.

Operating in character-mode (with 8-bits of data per character) rather than in pixel-mode graphics mode (with at least 8x12 bits of data per character) dramatically reduces the bandwidth required for the input-output communication between the computer and operator terminal.

3.1.2 Character-mode Graphical-style User Interface

The Linux "Terminal" application, user selectable terminal emulators (including multi-color xterms and non-color vt100/vt220) and "ncurses" terminal device interface library supports the TeamSTARS "tsWxGTUI_PyVx" Toolkit with its "wxPython"-style, "Curses"-based Graphical-Text User Interface (GUI).

The existing Character-mode Command Line Interface enables the operator to concurrently conduct not only one or more local GUI sessions but also one or more remote GUI sessions.

Operating in character-mode (with 8-bits of data per color-pair code) rather than in pixel-mode graphics mode (with at least 8x12x24 bits of data per character) dramatically reduces the bandwidth required for the input-output communication between the computer and operator terminal.

3.2 Mac OS X (POSIX-compatible CLI & GUI)

From Wikipedia, the free encyclopedia

"OS X, formerly known as Mac OS X, is a series of Unix-based graphical interface operating systems developed and marketed by Apple Inc. It is designed to run on Mac computers, having been pre-installed on all Macs since 2002.

Within the market of desktop, laptop and home computers, and by web usage, OS X is the second most widely used OS after Windows.

OS X, whose X is the Roman numeral for 10 and is a prominent part of its brand identity, is built on technologies developed at NeXT between the second half of the 1980s and Apple's purchase of the company in late 1996. The 'X' is also used to emphasize the relatedness between OS X and UNIX. Versions 10.5 "Leopard" running on Intel processors, 10.6 "Snow Leopard", 10.7 "Lion", 10.8 "Mountain Lion", 10.9 "Mavericks", and 10.10 "Yosemite" have obtained UNIX 03 certification.

iOS, which runs on the iPhone, iPod Touch, iPad, and the 2nd and 3rd generation Apple TV, shares the Darwin core and many frameworks with OS X. An unnamed variant of v10.4 powered the first generation Apple TV.

Early versions of Mac OS X were compiled to run on the PowerPC CPUs used by Macs of the period. After Apple announced it would shift to using Intel x86 CPUs from 2006 onwards, Tiger and Leopard were released in versions for Intel and PowerPC processors. Snow Leopard was the first version released only for Intel Macs. Since the release of Mac OS X 10.7 "Lion", OS X has dropped support for 32-bit Intel processors as well. It now runs exclusively on 64-bit Intel CPUs.

OS X is based upon the Mach kernel, developed at Carnegie Mellon University. Certain parts of FreeBSD's and NetBSD's were also incorporated into NeXTSTEP, which forms the core of Mac OS X."

3.2.1 Character-mode Command Line Interface

The Mac OS X "Terminal" or Third-party "iTerm" application and "bash" shell support the TeamSTARS "tsWxGTUI_PyVx" Toolkit with its Python-based Command Line Interface (CLI).

The "bash" shell, with its "ssh" and "sftp" commands, enables the operator to conduct local and remote sessions.

3.2.2 Character-mode Graphical-style User Interface

The third-party "iTerm" Mac OS X application, user selectable terminal emulators (including multi-color xterms and non-color vt100/vt220) and "ncurses" terminal device interface library supports the TeamSTARS "tsWxGTUI_PyVx" Toolkit with its "wxPython"-style, "Curses"-based Graphical-Text User Interface (GUI).

NOTES:

- a) The Mac OS X "Terminal" application, user selectable terminal emulators (including multi-color xterms and non-color vt100/vt220) and "ncurses" terminal device interface library is NOT recommended because it does not support operator input via a mouse.
- b) However, the TeamSTARS "tsWxGTUI_PyVx" Toolkit would be suitable for use with VT100 and VT220 terminal emulators because the Digital Equipment Corporation never designed its VT100 and VT220 terminals to work with operator input via a mouse. It only supported operator input via keyboard and the available function keys.

3.3 Microsoft Windows

From Wikipedia, the free encyclopedia

"Microsoft Windows or Windows is a metafamily of graphical operating systems developed, marketed, and sold by Microsoft. It consists of several families of operating systems, each of which cater to a certain sector of the computing industry. Active Windows families include Windows NT, Windows Embedded and Windows Phone; these may encompass subfamilies, e.g. Windows Embedded Compact (Windows CE) or Windows Server. Defunct Windows families include Windows 9x and Windows Mobile.

Microsoft introduced an operating environment named Windows on November 20, 1985 as a graphical operating system shell for MS-DOS in response to the growing interest in graphical user interfaces (GUIs). Microsoft Windows came to dominate the world's personal computer market with over 90% market share, overtaking Mac OS, which had been introduced in 1984. However, it is outsold by

Android on smartphones and tablets.

As of April 2014, the most recent versions of Windows for personal computers, smartphones, server computers and embedded devices are respectively Windows 8.1, Windows Phone 8.1, Windows Server 2012 R2 and Windows Embedded 8. A specialized version of Windows runs on the Xbox One game console."

NOTE: You will have to obtain third-party software in order to use various POSIX-compatible local and remote Command Line Interface and "Curses"-based Graphical-style User Interface features.

3.3.1 "Cygwin" (POSIX-compatible CLI & GUI)

From Wikipedia, the free encyclopedia

"Cygwin is a Unix-like environment and command-line interface for Microsoft Windows. Cygwin provides native integration of Windows-based applications, data, and other system resources with applications, software tools, and data of the Unix-like environment. Thus it is possible to launch Windows applications from the Cygwin environment, as well as to use Cygwin tools and applications within the Windows operating context.

Cygwin consists of two parts: a dynamic-link library (DLL) as an API compatibility layer providing a substantial part of the POSIX API functionality, and an extensive collection of software tools and applications that provide a Unix-like look and feel.

Cygwin was originally developed by Cygnus Solutions, which was later acquired by Red Hat. It is free and open source software, released under the GNU General Public License version 3. Today it is maintained by employees of Red Hat, NetApp and many other volunteers.

Cygwin consists of a library that implements the POSIX system call API in terms of Win32 system calls, a GNU development toolchain (including GCC and GDB) to allow software development, and a large number of application programs equivalent to those on Unix systems. Programmers have ported many Unix, GNU, BSD and Linux programs and packages to Cygwin, including the X Window System, K Desktop Environment 3, GNOME, Apache, and TeX. Cygwin permits installing inetd, syslogd, sshd, Apache, and other daemons as standard

Windows services, allowing Microsoft Windows systems to emulate Unix and Linux servers.

Cygwin programs are installed by running Cygwin's "setup" program, which downloads the necessary program and feature package files from repositories on the Internet. Setup can install, update, and remove programs and their source code packages. A complete installation will take in excess of 17 GB of hard disk space, but usable configurations may require as little as 1 or 2 GB."

3.3.1.1 Character-mode Command Line Interface

The Cygwin "Terminal" application and "bash" shell support the TeamSTARS "tsWxGTUI_PyVx" Toolkit with its Python-based Command Line Interface (CLI).

The "bash" shell, with its "ssh" and "sftp" commands, enables the operator to conduct local and remote sessions.

3.3.1.2 Character-mode Graphical-style User Interface

The Cygwin "Terminal" application, user selectable terminal emulators (including multi-color xterms and non-color vt100/vt220) and "ncurses" terminal device interface library supports the TeamSTARS "tsWxGTUI_PyVx" Toolkit with its "wxPython"-style, "Curses"-based Graphical-Text User Interface (GUI).

3.3.2 "GnuWin32" (POSIX-compatible CLI only)

From Wikipedia, the free encyclopedia

"The GnuWin32 project provides native ports in the form of runnable computer programs, patches, and source code for various GNU and open source tools and software, much of it modified to run on the 32-bit Windows platform. The ports included in the GnuWin32 packages are:

- * GNU utilities such as bc, bison, chess, Coreutils, diffutils, ed, Flex, gawk, gettext, grep, Groff, gzip, iconv, less, m4, patch, readline, rx, sharutils, sed, tar, texinfo, units, Wget, which
- * Archive management and compression tools, such as: arc, arj, bzip2, gzip, lha, zip, zlib.
- * Non-GNU utilities such as: cygutils, file, ntfsprogs, OpenSSL, PCRE.

- * Graphics tools.
- * PDCurses [Release 3.4]
- * Tools for processing text.
- * Mathematical software and statistics Software.

Most programs have dependencies (typically DLLs), so that the executable files cannot simply be run in Windows unless files they depend upon are available. An alternative set of ported programs is UnxUtils; these versions only depend on the Microsoft C-runtime msvcrt.dll, but are usually older versions."

NOTE: PDCurses does not provide the "ncurses" mouse button interface used by the Team-STARS "tsWxGTUI_PyVx" Toolkit. It also does not provide or recognize the xterm and vt100 terminal emulators.

Unfortunately, Python Curses for Microsoft Windows traps when it fails to import "_curses". Furthermore, the "tsWxGTUI" Toolkit will trap when it fails to obtain mouse button key codes from PDCurses 3.4 via these standard Python Curses module references:

```
curses.BUTTON1_RELEASED
curses.BUTTON1_PRESSED
curses.BUTTON1_CLICKED
curses.BUTTON1_DOUBLE_CLICKED
curses.BUTTON1_TRIPLE_CLICKED
curses.BUTTON2_RELEASED
curses.BUTTON2_PRESSED
curses.BUTTON2_CLICKED
curses.BUTTON2_DOUBLE_CLICKED
curses.BUTTON2_TRIPLE_CLICKED
curses.BUTTON3_RELEASED
curses.BUTTON3_PRESSED
curses.BUTTON3_CLICKED
curses.BUTTON3_DOUBLE_CLICKED
curses.BUTTON3_TRIPLE_CLICKED
curses.BUTTON4_RELEASED
curses.BUTTON4_PRESSED
curses.BUTTON4_CLICKED
curses.BUTTON4_DOUBLE_CLICKED
curses.BUTTON4_TRIPLE_CLICKED
curses.BUTTON_CTRL
curses.BUTTON_SHIFT
curses.BUTTON_ALT
```

3.3.2.1 Character-mode Command Line Interface

The GnuWin32 "Terminal" application and "bash" shell support the TeamSTARS "tsWxGTUI_PyVx" Toolkit with its Python-based Command Line Interface (CLI).

The "bash" shell, with its "ssh" and "sftp" commands, enables the operator to conduct local and remote sessions.

3.3.2.2 Character-mode Graphical-style User Interface (Availability depends on PDCurses)

The GnuWin32 "Terminal" application, user selectable terminal emulators (including multi-color xterms and non-color vt100/vt220) and "PDCurses" terminal device interface library supports the TeamSTARS "tsWxGTUI_PyVx" Toolkit with its "wxPython"-style, "PDCurses"-based Graphical-Text User Interface (GUI).

3.3.3 "Windows PowerShell" (Windows-native CLI only)

From Wikipedia, the free encyclopedia

"Windows PowerShell is a task automation and configuration management framework from Microsoft, consisting of a command-line shell and associated scripting language built on the .NET Framework. PowerShell provides full access to COM and WMI, enabling administrators to perform administrative tasks on both local and remote Windows systems as well as WS-Management and CIM enabling management of remote Linux systems and network devices.

In PowerShell, administrative tasks are generally performed by cmdlets (pronounced command-lets), which are specialized .NET classes implementing a particular operation. Sets of cmdlets may be combined into scripts, executables (which are standalone applications), or by instantiating regular .NET classes (or WMI/COM Objects). These work by accessing data in different data stores, like the file system or registry, which are made available to the PowerShell runtime via Windows PowerShell providers.

Windows PowerShell also provides a hosting API with which the Windows PowerShell runtime can be embedded inside other applications. These applications can then use Windows PowerShell functionality to implement certain operations, including those exposed via the graphical

interface. This capability has been used by Microsoft Exchange Server 2007 to expose its management functionality as PowerShell cmdlets and providers and implement the graphical management tools as PowerShell hosts which invoke the necessary cmdlets. Other Microsoft applications including Microsoft SQL Server 2008 also expose their management interface via PowerShell cmdlets. With PowerShell, graphical interface-based management applications on Windows are layered on top of Windows PowerShell. A PowerShell scripting interface for Windows products is mandated by Microsoft's Common Engineering Criteria.

Windows PowerShell includes its own extensive, console-based help, similar to man pages in Unix shells, via the Get-Help cmdlet and updatable with fresh content using the Update-Help cmdlet and web based content via the -online switch to Get-Help."

3.3.3.1 Character-mode Command Line Interface

The "PowerShell" application supports the TeamSTARS "tsWxGTUI_PyVx" Toolkit with its Python-based Command Line Interface (CLI).

3.3.3.2 Character-mode Graphical-style User Interface

Not supported by the TeamSTARS "tsWxGTUI_PyVx" Toolkit.

3.3.4 "Command Prompt" (Windows-native CLI only)

From Wikipedia, the free encyclopedia

"Command Prompt, better known as cmd.exe or just cmd (after its executable file name), is the command-line interpreter on OS/2 and eComStation, Windows CE and Windows NT operating systems (including Windows 2000 and later). It is the analog of COMMAND.COM in DOS and Windows 9x systems (where it is also called "MS-DOS Prompt"), or of the Unix shells used on Unix-like systems.

Unlike COMMAND.COM, which is a DOS program, cmd is a native Windows application usually running in Win32 console. This allows it to take advantage of features available to native programs on the platform that are otherwise unavailable to DOS programs. For example, since cmd is a native text mode application on OS/2, it can use real pipes in command pipelines, allowing both sides of the

pipeline to run concurrently. As a result, it is possible to redirect the standard error in cmd, unlike COMMAND.COM. (COMMAND.COM uses temporary files, and runs the two sides serially, one after the other.)

In reality, cmd is a Windows program that acts as a DOS-like command line interpreter. It is generally compatible, but provides extensions which address some of the limitations of COMMAND.COM."

For details, please see:

<http://en.wikipedia.org/wiki/Cmd.exe>

3.3.4.1 Character-mode Command Line Interface

The "Command Prompt" application supports the TeamSTARS "tsWxGTUI_PyVx" Toolkit with its Python-based Command Line Interface (CLI).

3.3.4.2 Character-mode Graphical-style User Interface

Not supported by the TeamSTARS "tsWxGTUI_PyVx" Toolkit.

3.4 Unix (POSIX-compatible CLI & GUI)

From Wikipedia, the free encyclopedia

"Unix (all-caps UNIX for the trademark) is a family of multitasking, multiuser computer operating systems that derive from the original AT&T Unix, developed in the 1970s at the Bell Labs research center by Ken Thompson, Dennis Ritchie, and others.

Initially intended for use inside the Bell System, AT&T licensed Unix to outside parties from the late 1970s, leading to a variety of both academic and commercial variants of Unix from vendors such as the University of California, Berkeley (BSD), Microsoft (Xenix), IBM (AIX) and Sun Microsystems (Solaris). AT&T finally sold its rights in Unix to Novell in the early 1990s, which then sold its Unix business to the Santa Cruz Operation (SCO) in 1995, but the UNIX trademark passed to the industry standards consortium The Open Group, which allows the use of the mark for certified operating systems compliant with the Single UNIX Specification. Among these is Apple's OS X, which is the Unix version with the largest installed base as of 2014.

From the power user's or programmer's perspective, Unix systems are characterized by a modular design that is sometimes called the "Unix philosophy," meaning the OS provides a set of simple tools that each perform a

limited, well-defined function, with a unified file-system as the main means of communication and a shell scripting and command language to combine the tools to perform complex workflows. Aside from the modular design, Unix also distinguishes itself from its predecessors as the first portable operating system: virtually the entire OS is written in the C programming language which allowed it to outgrow the 16-bit PDP-11 minicomputer for which it was originally developed.

Many clones of Unix have arisen over the years, of which Linux is the most popular, having overtaken the popularity of "true" Unix on server platforms since its inception in the early 1990s.

Originally, Unix was meant to be a programmer's workbench to be used for developing software to be run on multiple platforms more than to be used to run application software. The system grew larger as the operating system started spreading in the academic circle, as users added their own tools to the system and shared them with colleagues.

Unix was designed to be portable, multi-tasking and multi-user in a time-sharing configuration. Unix systems are characterized by various concepts: the use of plain text for storing data; a hierarchical file system; treating devices and certain types of inter-process communication (IPC) as files; and the use of a large number of software tools, small programs that can be strung together through a command line interpreter using pipes, as opposed to using a single monolithic program that includes all of the same functionality. These concepts are collectively known as the "Unix philosophy." Brian Kernighan and Rob Pike summarize this in *The Unix Programming Environment* as "the idea that the power of a system comes more from the relationships among programs than from the programs themselves."

Unix operating systems are widely used in servers, workstations, and mobile devices.[11] The Unix environment and the client-server program model were essential elements in the development of the Internet and the reshaping of computing as centered in networks rather than in individual computers.

Both Unix and the C programming language were developed by AT&T and distributed to government and academic institutions, which led to both being ported to a wider variety of machine families than any other operating system.

Under Unix, the operating system consists of many utilities along with the master control program, the

kernel. The kernel provides services to start and stop programs, handles the file system and other common "low level" tasks that most programs share, and schedules access to avoid conflicts when programs try to access the same resource or device simultaneously. To mediate such access, the kernel has special rights, reflected in the division between user-space and kernel-space.

The microkernel concept was introduced in an effort to reverse the trend towards larger kernels and return to a system in which most tasks were completed by smaller utilities. In an era when a standard computer consisted of a hard disk for storage and a data terminal for input and output (I/O), the Unix file model worked quite well, as most I/O was linear. However, modern systems include networking and other new devices. As graphical user interfaces developed, the file model proved inadequate to the task of handling asynchronous events such as those generated by a mouse. In the 1980s, non-blocking I/O and the set of inter-process communication mechanisms were augmented with Unix domain sockets, shared memory, message queues, and semaphores. In microkernel implementations, functions such as network protocols could be moved out of the kernel, while conventional (monolithic) Unix implementations have network protocol stacks as part of the kernel."

3.4.1 Character-mode Command Line Interface

The Unix "Terminal" application and "bash" shell support the TeamSTARS "tsWxGTUI_PyVx" Toolkit with its Python-based Command Line Interface (CLI).

The "bash" shell, with its "ssh" and "sftp" commands, enables the operator to conduct local and remote sessions.

3.4.2 Character-mode Graphical-style User Interface

The Unix "Terminal" application, user selectable terminal emulators (including multi-color xterms and non-color vt100/vt220) and "ncurses" terminal device interface library supports the TeamSTARS "tsWxGTUI_PyVx" Toolkit with its "wxPython"-style, "Curses"-based Graphical-Text User Interface (GUI).

===== TOOLKIT DEVELOPMENT RESOURCES =====

4. Toolkit Development Resources

The TeamSTARS "tsWxGTUI_PyVx" Toolkit is designed to be used without modification on computer systems with 32-bit and 64-bit processors from various manufacturers.

4.1 Python Programming Language Resources

The TeamSTARS "tsWxGTUI_PyVx" Toolkit is implemented in both of the currently popular, high-level Python programming languages.

Both versions are generally available for computer systems with various editions of the Linux, Mac OS X, Microsoft Windows and Unix operating systems.

4.1.1 Python 3.x

Stable releases of evolving Python 3x version 3.0.0-3.4.3 are available for use with current and future computer platforms.

Latest release candidate is Python 3.5.0rc2 as of 9 Mar. 2015.

NOTE: The Python Software Foundation has designated Python 3.x to be under active development.

There will be ongoing feature enhancement upgrades.

There will be a limited number of bug fix updates to earlier Python 3.x releases.

4.1.2 Python 2.x

Stable releases of mature Python 2x versions 2.0.0-2.7.9 are available for use with legacy computer platforms.

NOTE: The Python Software Foundation has designated Python 2.x to be in its End-Of-Life stage.

There will be no more feature enhancement upgrades (back-ported from Python 3x).

There will be a limited number of bug fix updates to Python 2.7.

4.2 Open Source & Commercial Python Resources

Microsoft Windows operating systems do not include any Python release.

Linux, Mac OS X and Unix operating systems typically include a recent, but not necessarily the latest Python release.

The following open source and commercial resources can provide Python releases to suite your current and evolving needs.

4.2.1 Python Software Foundation Website

From <http://www.python.org/>

"Python is a programming language that lets you work quickly and integrate systems more effectively."

Python runs on Linux, Mac OS X, Microsoft Windows, Unix and other Operating Systems. It also has been ported to the Java and .NET virtual machines.

Python is free to use, even for commercial products, because of its OSI-approved open source license.

New to Python or choosing between Python 2 and Python 3? Read Python 2 or Python 3.

The Python Software Foundation holds the intellectual property rights behind Python, underwrites the PyCon conference, and funds many other projects in the Python community.

The current production versions are Python 2.7.9 and Python 3.4.3.

Start with one of these versions for learning Python if you want the most stability; they're both considered stable production releases.

If you don't know which version to use, try Python 3.4.x. Some existing third-party software is not yet compatible with Python 3; if you need to use such software, you can download Python 2.7.9 instead."

4.2.2 ActiveState Website

From <http://www.activestate.com/activepython:>

"ActivePython Business and Enterprise Editions feature our precompiled, supported, quality-assured Python distribution used by millions of developers around the world for easy Python installation and quality-assured code. When you're using Python on production servers or mission-critical applications, ActivePython Business and Enterprise Editions offer significant time savings over open source Python for installing, managing, and standardizing your Python."

"ActivePython (freeware, Windows)
 IDLE (open source, Windows)
 Komodo (commercial, Windows)
 RPMs (open source, Linux)
 Source code (all platforms)
 python-mode for Emacs
 Jack Jansen's binaries (open source, Mac OS, Mac OS X)
 Fink (open source, Mac OS X)
 Jython (open source, Java)
 more platforms"

4.3 Python Training Resources

4.3.1 Dive into Python

From <http://www.diveintopython.net/>

"Dive Into Python is a free Python book for experienced programmers. It was originally hosted at DiveIntoPython.org, but the author has pulled down all copies. It is being mirrored here. You can read the book online, or download it in a variety of formats. It is also available in multiple languages."

4.3.2 Learn Python

"Learning to Program How to Think Like a Computer Scientist"

Thinking in Python
 Introductions
 O'Reilly Python Center
 Non-English resources
 more tutorials"

4.3.3 Python Community

"Python.org
 comp.lang.python
 c.l.p.announce
 Tutor
 Python mailing lists
 more links"

4.3.4 Python Cookbook

"Python Cookbook
 Vaults of Parnassus
 Python.facts"

===== PYTHON DOWNLOAD GOTCHAS =====

5. Python Download Gotchas

How to check if python is already installed on your computer?

Read "<https://wiki.python.org/moin/BeginnersGuide/Download>"

For those Operating System distributions that don't include Python, various release versions can be obtained directly from the Python Software Foundation at "www.python.org".

5.1 Linux and Unix Operating Systems (CAUTION)

Distributions of Linux and Unix Operating Systems typically include a recent, but not necessarily the latest, Python release. However, since the Operating System distribution may itself use a specific Python release, it is often necessary to follow special installation procedures when adding newer Python releases.

For example, see the "How to install Python 2.7 and Python 3.3 on CentOS 6" article at:

<http://toomuchdata.com>.

It advises its readers to:

"Use 'make altinstall' to prevent problems.

It is critical that you use make altinstall when you install your custom version of Python. If you use the normal make install you will end up with two different versions of Python in the filesystem both named python. This can lead to problems that are very hard to diagnose."

5.2 Microsoft Windows Operating Systems (WARNING)

Distributions of Microsoft Windows do NOT include any Python release and do NOT include any "Curses" or "nCurses" Terminal Control Library for use with the terminal emulators (xterm, xterm-color, xterm-16color, xterm-88color, xterm-256color, vt100 and vt220) and terminals.

However, if you install Cygwin, the free Linux-like Command Line Interface and GNU Toolkit from Red Hat at "<http://cygwin.com>", you can then install a recent, but not necessarily the latest Python 2.x and/or Python 3.x release and the associated "Curses" or "nCurses" Terminal Control Library.

===== WXPYTHON/WXWIDGETS DEVELOPMENT RESOURCES =====

6. wxPython/wxWidgets Development Resources

The TeamSTARS "tsWxGTUI_PyVx" Toolkit began, in 2007, as a character-mode emulation of the Application Programming Interface (API) of the pixel-mode wxPython 2.8.9.2 GUI Toolkit used by Python language programmers.

By contrast, the pixel-mode wxPython GUI Toolkit is only a wrapper which interfaces Python language application programs to the pixel-mode wxWidgets GUI Toolkit components which are themselves implemented in the C++ programming language.

In October 2014, the wxWidgets and wxPython developers released version 3.0.2.0 and associated on-line documentation. While documentation for new and older wxWidgets releases is available, the documentation for wxPython 2.8.9.2 is no longer available and has been replaced by links to wxWidgets documentation. Documentation for wxPython 2.8.9.2 has been replaced by a link to wxWidgets 2.8.12. The new documentation lacks diagrams for each class which depict the class inheritance relationships.

6.1 wxPython Programming Language Resources

6.1.1 wxPython 2.8.9.2 (Currently Emulated API)

6.1.2 wxPython 3.0.2.0 (Future Emulated API)

6.2 Open Source wxPython/wxWidgets Resources

6.2.1 wxPython.org Website

6.2.2 wxWidgets.org Website

6.3 wxPython/wxWidgets Training Resources

6.3.1 wxPython Community

6.3.2 wxPython Cookbook

<http://wiki.wxpython.org>

===== End-Of-File =====

8.1.1 System Requirements

8.1.1.1 Hardware

Hardware components must include application-specific versions of the following:

8.1.1.1.1 Central Processing Unit

A required electronic device that loads and executes machine-readable instructions.

It performs arithmetic and logic. It loads, modifies and stores machine-readable data. The processor may be any of the popular 32-/64-bit single or multi-core devices.

8.1.1.1.2 Random Access Memory

A required electronic device that temporarily receives, stores and returns machine-readable machine instructions and data that will be available until the next power interruption.

8.1.1.1.3 Non-Volatile Storage

A required internal and optional external electronic device (Flash Memory) or electro-mechanical device (Hard Drive) whose non-volatile memory receives, stores and can then return source and machine-readable data after electrical power interruptions. The optional external device may be local or remote. When remote, the system must include a Network Interface Device.

8.1.1.1.4 Network Interface Device

An optional electronic device (Modem) that inputs and outputs data over an optional wired or wireless telecommunications circuit.

8.1.1.1.5 Keyboard Device

A required hand-operated electronic or electro-mechanical device to input alpha-numeric and control data via an assembly of typewriter-style push buttons.

8.1.1.1.6 Pointing Device

For the Graphical User Interface (GUI) mode of operation, this is a required hand-operated electronic or electro-mechanical Mouse, Trackball, Touchpad or Touchscreen device that controls the coordinates of a cursor on the computer screen as you move the positioning control (hand, finger or stylus) around.

The pointing device detects two-dimensional motion relative to the surface of the computer's display. This motion is typically translated into the motion of a pointer (cursor) on the display, which enables accurate operations involving GUI objects (such as editing text and controlling buttons, checkboxes, scrollbars, sliders etc.).

The most common standard mouse features:

1 For Linux, Microsoft Windows and Unix:

- a) two button (left & right) mouse
- b) a scroll wheel, which can also act as a third (middle) button
- c) an optional touchpad or touchscreen with software that can recognize one and two finger gestures such as tap, drag and scroll

2 For Mac OS X:

- a) a single button mouse
- b) an optional touchpad with software that can recognize one and two finger gestures such as tap, drag and scroll

8.1.1.1.7 Display Device

A required electronic device that outputs alpha-numeric, graphic (or graphic-style) and control data to the computer screen.

1 Basic Computer Terminal Display and Keyboard for Software Engineering Workstation

A 12" pixel-mode multi-font graphics display that supports at least 80 col x 40 row (640 x 480 pixels) and 16,777,216 colors and industry standard terminal emulators.

- a) xterm (8-color, 64-color pairs)
- b) xterm-16color (16-color, 256-color-pairs)
- c) vt100 (1-color ON/OFF, 2-color-pair NORMAL/REVERSE)
- d) vt220 (1-color ON/OFF, 2-color-pair NORMAL/REVERSE)

2 Basic Computer Terminal Display and Keyboard for Embedded System Operator

Applications may be as simple as a single frame window. A frame requires: 1) a title line with optional window control buttons; and 2) additional optional lines for a menu bar, tool bar, status bar, buttons, check boxes, radio boxes & buttons, gauges and scrolled text. The display size for this is 35 col x 16 row (280 x 200 pixels).

More complex applications involve multiple side-by-side and overlapping frames which may optionally be arranged in a desktop consisting of: 1) a collection of application frame and dialog windows; 2) a scrolling operator event notification window ; and 3) a taskbar whose buttons enable the operator to raise hidden frames and dialogues from the invisible background to the visible foreground. For the sample splash screen, the display size can range from 60 col x 25 row (480 x 300 pixels) to over 80 col x 50 row (640 x 600 pixels).

A graphic and/or character-mode single font text display (whose size could be somewhere between that of a 3" smart phone and a 12" tablet) that supports the application (or its splash screen) and industry standard terminal emulators:

- a) xterm (8-color, 64-color pairs)
- b) xterm-16color (16-color, 256-color-pairs)
- c) vt100 (1-color ON/OFF, 2-color-pair NORMAL/REVERSE)
- d) vt220 (1-color ON/OFF, 2-color-pair NORMAL/REVERSE)

3 Optional Computer Terminal Display for Software Engineering Workstation

Any one of the following may be substituted for the **Basic Computer Terminal Display** based on your need to simultaneously view multiple software engineering documents and activities.

Derived From Wikipedia, the free encyclopedia at "https://en.wikipedia.org/wiki/Display_resolution"

Acronym (usage)	Aspect ratio	Width (pixels)	Height (pixels)
VGA (12" CRT)	4:3	640	480
SVGA (14" CRT)	4:3	800	600
XGA (15" Laptop)	4:3	1024	768

XGA+	4:3	1152	864
WXGA	16:9	1280	720
WXGA	5:3	1280	768
WXGA	16:10	1280	800
SXGA– (UVGA)	4:3	1280	960
SXGA	5:4	1280	1024
HD	~16:9	1360	768
HD	~16:9	1366	768
SXGA+	4:3	1400	1050
WXGA+ (17" Laptop)	16:10	1440	900
HD+	16:9	1600	900
UXGA	4:3	1600	1200
WSXGA+	16:10	1680	1050
FHD	16:9	1920	1080
WUXGA	16:10	1920	1200
QWXGA	16:9	2048	1152
WQHD (27" Desktop)	16:9	2560	1440
WQXGA	16:10	2560	1600

8.1.1.1.8 Printer Device

An optional electro-mechanical device that outputs alpha-numeric, graphic and control data to individual sheets of paper.

8.1.1.2 Software

Software components must include application-specific versions of the following:

8.1.1.2.1 Operating System

Platform-specific, multi-user (enables a user to login locally and the same or another user to login remotely), multi-process (enables a user to launch multiple applications and/or multiple instances of the same application), multi-threaded (enables each application to concurrently execute multiple tasks which may independently block while waiting for a triggering event notification) Operating Systems for 32-bit/64-bit processors:

- Linux (such as CentOS, Debian GNU, Fedora, Gentoo, Hard Hat, Mandriva, OpenSuSE, Red Hat, Scientific, Ubuntu, Yellow Dog etc.).
- Mac OS X (such as Unix-like Darwin and BSD based 10.4/Tiger - 10.10/Yosemite etc.)
- Microsoft Windows (such as Professional or Home editions of 8.1, 8, 7, Vista or XP) when augmented with Cygwin, the free, Linux-like Command Line Interface and GNU toolkit from Red Hat.
- Unix (such as FreeBSD/PC-BSD, HP-UX, IBM-AIX, IRIX, OpenIndiana, SCO, Solaris/SunOS etc.).

8.1.1.2.2 Command Line Interface Shell

Host Computer Operating System Command Line Interface Shell.

- Linux "Terminal".
- Mac OS X "iTerm", "iTerm2" or "Terminal".
- Microsoft Windows Command Prompt "cmd.exe" or "PowerShell.exe".
- Unix "Terminal".

8.1.1.2.3 Graphical-style User Interface Shell

Host Computer Operating System Graphical-style User Interface Shell.

- Linux "GNOME", "KDE", "X11" Desktop Window Manager.
- Mac OS X Desktop Window Manager.
- Microsoft Windows Desktop Window Manager.
- Unix "GNOME", "KDE", "X11" Desktop Window Manager.

8.1.1.2.4 Terminal Interface Libraries

Excerpts from From Wikipedia, the free encyclopedia at

[https://en.wikipedia.org/wiki/Curses_\(programming_library\)](https://en.wikipedia.org/wiki/Curses_(programming_library))

"curses is a terminal control library for Unix-like systems, enabling the construction of text user interface (TUI) applications.

The name is a pun on the term "cursor optimization". It is a library of functions that manage an application's display on character-cell terminals (e.g., VT100).

Overview

The curses API is described in several places. Most implementations of curses use a database that can describe the capabilities of thousands of different terminals. There are a few implementations, such as PDCurses, which use specialized device drivers rather than a terminal database. Most implementations use terminfo; some use termcap. Curses has the advantage of back-portability to character-cell terminals and simplicity. For an application that does not require bit-mapped graphics or multiple fonts, an interface implementation using curses will usually be much simpler and faster than one using an X toolkit.

Using curses, programmers are able to write text-based applications without writing directly for any specific terminal type. The curses library on the executing system sends the correct control characters based on the terminal type. It provides an abstraction of one or more windows that maps onto the terminal screen. Each window is represented by a character matrix. The programmer sets up each window to look as they want the display to look, and then tells the curses package to update the screen. The library determines a minimal set of changes needed to update the display and then executes these using the terminal's specific capabilities and control sequences.

In short, this means that the programmer simply creates a character matrix of how the screen should look and lets curses handle the work."

"Portability

Although the ncurses library was initially developed under Linux, OpenBSD, FreeBSD, and NetBSD it has been ported to many other ANSI/POSIX UNIX systems, mainly by Thomas Dickey. PDCurses, while not identical to ncurses, uses the same function calls and operates the same way as ncurses does except that PDCurses targets different devices, e.g., console windows for DOS, Win32, OS/2, as well as X11. Porting between the two is not difficult. For example, the roguelike game ADOM was written for Linux and ncurses, later ported to DOS and PDCurses."

Host Computer Terminal Interface Libraries.

- Linux "Curses"/"nCurses" Terminal Interface Library.
- Mac OS X "Curses"/"nCurses" Terminal Interface Library.
- Microsoft Windows "Curses"/"nCurses" Terminal Interface Library provided by Cygwin, the free, Linux-like Command Line Interface and GNU toolkit from Red Hat.
- Unix "Curses"/"nCurses" Terminal Interface Library.

8.1.1.2.5 Terminal Emulators

Host Computer Terminal Emulators.

- Linux vt100, vt220, xterm, xterm-color, xterm-16color, xterm-88color and xterm-256color.
- Mac OS X vt100, vt220, xterm, xterm-color, xterm-16color, xterm-88color and xterm-256color.
- Microsoft Windows vt100, vt220, mintty, xterm, xterm-color, xterm-16color, xterm-88color and xterm-256color provided by Cygwin, the free, Linux-like Command Line Interface and GNU toolkit from Red Hat.
- Unix vt100, vt220, xterm, xterm-color, xterm-16color, xterm-88color and xterm-256color.

8.1.1.2.6 Development Tools

Host Computer Development Tools:

- File Manager (such as "Midnight Commander" on Linux and Unix; "Finder" and "PathFinder" on Mac OS X; "File Explorer" and "Total Commander" on Microsoft Windows etc.).
- Text (source code) editor (such as "vi", "xemacs" etc.).
- Optional Word Processor (source code) editor (such as "Microsoft Word", "AuthorIt", "LibreOffice Text Document" etc.).
- Optional Block Diagram and Flowchart editor (such as "Microsoft Visio", "LibreOffice Presentation", "LibreOffice Drawing" etc.).
- Optional Database (such as "Microsoft Access", "MySQL", "PostgreSQL", "SQLite", "LibreOffice Database" etc.).
- Optional Spreadsheet (such as "Microsoft Excel", "LibreOffice Spreadsheet" etc.).
- Optional Programm debugger (such as "gdb").

8.1.1.2.7 Python Virtual Machines

Python Virtual Machine and its associated Python source code compiler and byte code interpreter.

8.1.1.2.8 Python Integrated Development Environments

Python Integrated Development Environments (such as "Python Idle", WingWare's "WingIDE" etc.)

8.1.1.2.9 TeamSTARS "tsWxGTUI" Toolkit

The "tsWxGTUI" Toolkit and its associated library of general purpose, re-usable buildig block modules, developmnt tools and utilities.

8.1.1.3 Baseline Development and Test Platforms

The following Development and Test Platforms represent a broad spectrum of Intel microprocessor based systems that are known to support the TeamSTARS "tsWxGTUI" Toolkit.

NOTE: Since cross-platform operating system and Python virtual machine technology is also available for non-Intel based systems, it is likely that the *TeamSTARS* "tsWxGTUI" toolkit will also work on those systems which use the equivalent operating systems and Python virtual machines with 32-bit and 64-bit microprocessors from other manufacturers including:

- AMD
- ARM Holdings
- Cyrix
- Freescale
- Intel
- IBM
- Marvell
- NexGen
- Nvidia Tegra
- Oracle (previously Sun)
- OWC
- Qualcomm
- Rise Technology
- Samsung
- SigmaTel
- Texas Instruments
- Transmeta
- tilera
- Via (Centaur Technology division)
- winchip

8.1.1.3.1 Low-Range Development & Operator Platform

1998-model year, 366 MHz Intel Pentium II-based laptop with 384 MB RAM, 640x480/1024x768 pixel resolution display and expansion capabilities (marginal resources and performance) sufficient enough to serve as the low-level baseline development and operator platform.

Its interchangeable 32 GB (4200 RPM) ATA hard drives were used to run Windows XP Pro or the Linux operating system kernel with GNU toolchain (Ubuntu 12.04).

Its limited memory and available PCMCIA network adapters were incompatible with later versions of Windows or with other Linux distributions.

8.1.1.3.2 Mid-Range Development & Operator Platform

2007-model year, 2.33 GHz Intel Core 2 Duo processor-based laptop with 4 GB RAM, 1920x1200 pixel resolution display and sufficient performance, resources and expansion capabilities to serve as the moderate-level baseline development and operator platform.

Its 160 GB (5400 RPM) SATA 1.5 Gb/s internal hard drive was used to run Apple's Mac OS X 10-3-10.7 and the hypervisor virtualization applications (Parallels Desktop 3-8 and VMware Fusion 4-5) that supported various guest operating systems.

Its 1.5 TB (7200 RPM) SATA 3 Gb/s external hard drive was used to store and run configured versions of the following guest operating systems:

- Linux operating system kernel with GNU toolchain (Ubuntu 12.04).
- Microsoft Windows (8 Pro, 7 Pro and XP Pro).
- Unix (OpenSolaris 11/OpenIndiana 151a5).

8.1.1.3.3 High-Range Development & Operator Platform

2013-model year, 3.5 GHz Intel Quad Core i7 processor-based desktop with 16 GB RAM, 2560x1440 pixel resolution display and sufficient performance, resources and expansion capabilities to serve as the high-level baseline development and operator platform. Its 3 TB (7200 RPM) SATA 6 Gb/s internal hard drive had 128 GB Solid State Flash memory and was used to run Apple's Mac OS X 10.9-10.10 and the hypervisor virtualization applications (Parallels Desktop 9-10 and VMware Fusion 5) that supported various guest operating systems.

Its 3 TB (7200 RPM) SATA 6 Gb/s internal hard drive was also used to store and run configured versions of the following guest operating systems:

Linux operating system kernel with GNU tool-chain (CentOS 7.0, Fedora 20, OpenSUSE 13.1, Scientific 6.5 and Ubuntu 12.04 & 14.04);

Microsoft Windows operating system (8.1 Pro, 8 Pro, 7 Pro and XP Pro).

NOTE:

CLI features are available with Windows Command Prompt, Windows PowerShell or various third party shells.

GUI features become available to Windows users only after the users install Cygwin, the free and open source Linux-based add-on from Red Hat. Cygwin provides Windows users with POSIX-compatible command line shells, terminal control libraries, terminal emulators and the GNU toolchain).

Unix operating system (FreeBSD 9.2-10.0, OpenIndiana 151a8 PC-BSD 9.2-10.0).

8.1.1.4 Command Line Interface

User-friendly character-mode, cross-platform, Linux-/Unix-style processing of keyword-value pair options and positional arguments.

Each platform must provide the following building blocks in the Python version-specific Global Module Index:

- "argparse" (introduced with Python 2.7.0 and 3.2.0)
- "optparse" (introduced with Python 2.3.0 and 3.0.0; subsequently deprecated by argparse)
- "getopt" (introduced with Python 1.6.0 and 3.0.0)

8.1.1.5 Graphical User Interface

User-friendly character-mode, cross-platform, emulation of the pixel-mode "wxPython" GUI Toolkit.

Each platform must provide the following "wxPython"-style GUI-objects: frames, dialogs, menu bars, scroll bars, status bars, text bar, buttons, checkboxes, radio boxes/buttons, scrollable text and mouse cursor and buttons.

It must provide the following "wxPython"-style 68-color palette:

- a) No colors may appear in output for vt100 and vt200 displays;
- b) Only the standard 8-colors may appear in output for cygwin mintty, xterm and xterm-color displays based on color substitution mappings from the 68-color palette;
- c) Only the standard 16-colors may appear in output for xterm-16color displays based on color substitution mappings from a 71-color palette;
- d) If the configuration control parameter `USE_256_COLOR_PAIR_LIMIT` is set to True, then only the standard 16-colors may appear in output for xterm-88color displays, based on color substitution mappings from a 71-color palette;

If the configuration control parameter `USE_256_COLOR_PAIR_LIMIT` is set to False, then the experimental 71-color palette may be applied without need for any mapping;

- e) If the configuration control parameter `USE_256_COLOR_PAIR_LIMIT` is set to True, then only the standard 16-colors may appear in output for xterm-256color displays, based on color substitution mappings from a 71-color palette;

If the configuration control parameter `USE_256_COLOR_PAIR_LIMIT` is set to False, then the experimental 140-color palette may be applied without need for any mapping.

8.1.1.6 Terminal Control Library Interface

Excerpts From Wikipedia, the free encyclopedia at:

[https://en.wikipedia.org/wiki/Curses_\(programming_library\)](https://en.wikipedia.org/wiki/Curses_(programming_library))

"Overview

The curses API is described in several places. Most implementations of curses use a database that can describe the capabilities of thousands of different terminals. There are a few implementations, such as PDCurses, which use specialized device drivers rather than a terminal database. Most implementations use terminfo; some use termcap. Curses has the advantage of back-portability to character-cell terminals and simplicity. For an application that does not require bit-mapped graphics or multiple fonts, an interface implementation using curses will usually be much simpler and faster than one using an X toolkit.

Using curses, programmers are able to write text-based applications without writing directly for any specific terminal type. The curses library on the executing system sends the correct control characters based on the terminal type. It provides an abstraction of one or more windows that maps onto the terminal screen. Each window is represented by a character matrix. The programmer sets up each window to look as they want the display to look, and then tells the curses package to update the screen. The library determines a minimal set of changes needed to update the display and then executes these using the terminal's specific capabilities and control sequences.

In short, this means that the programmer simply creates a character matrix of how the screen should look and lets curses handle the work."

"Portability

Although the ncurses library was initially developed under Linux, OpenBSD, FreeBSD, and NetBSD it has been ported to many other ANSI/POSIX UNIX systems, mainly by Thomas Dickey. PDCurses, while not identical to ncurses, uses the same function calls and operates the same way as ncurses does except that PDCurses targets different devices, e.g., console windows for DOS, Win32, OS/2, as well as X11. Porting between the two is not difficult. For example, the roguelike game ADOM was written for Linux and ncurses, later ported to DOS and PDCurses."

- 1 Each platform shall provide a curses-style Terminal-independent Application Programming Interface (API).
 - a) Termcap-based curses on BSD UNIX
 - b) Terminfo-based curses on AT&T UNIX
 - c) Terminfo-based nCurses on Linux and on Cygwin for Microsoft Windows
- 2 Some platforms may optionally provide a PDCurses-base Terminal-independent Application Programming Interface (API).

8.1.1.7 Terminal Emulators

Each platform must provide a program that emulates a video terminal within some other display architecture. Though typically synonymous with a shell or text terminal, the term terminal covers all remote terminals, including graphical interfaces. A terminal emulator inside a graphical user interface is often called a terminal window.

- a) Mouse-supported (only with nCurses on Cygwin mintty and Linux XTerm/UXTerm), non-color (black and single shade of white, orange or green) for vt100 and vt220 displays;
- b) Mouse-supported 8-color (with support of 64 color pairs) for cygwin mintty, xterm and xterm-color displays;
- c) Mouse-supported 16-colors (with support of 256 color pairs) for xterm-16color displays;
- d) Mouse-supported 16-colors (with support of 256 color pairs) for xterm-88color displays, if the configuration control parameter `USE_256_COLOR_PAIR_LIMIT` is set to True; else Mouse-supported experimental 71-colors (with support of 5041 color pairs) for xterm-88color displays, if the configuration control parameter `USE_256_COLOR_PAIR_LIMIT` is set to False and the experimental 71-color palette will be applied without need for any mapping;
- e) Mouse-supported 16-colors (with support of 256 color pairs) for xterm-256color displays, if the configuration control parameter `USE_256_COLOR_PAIR_LIMIT` is set to True; else Mouse-supported experimental 140-colors (with support of 19600 color pairs) for xterm-256color displays, if the configuration control parameter `USE_256_COLOR_PAIR_LIMIT` is set to False and the experimental 140-color palette will be applied without need for any mapping.

8.1.2 Python Programming Resources

8.1.2.1 Distributions

8.1.2.1.1 Python Software Foundation Website

From <http://www.python.org/>

"Python is a programming language that lets you work quickly and integrate systems more effectively."

Python runs on Linux, Mac OS X, Microsoft Windows, Unix and other Operating Systems. It also has been ported to the Java and .NET virtual machines.

Python is free to use, even for commercial products, because of its OSI-approved open source license.

New to Python or choosing between Python 2 and Python 3? Read Python 2 or Python 3.

The Python Software Foundation holds the intellectual property rights behind Python, under-writes the PyCon conference, and funds many other projects in the Python community.

The current production versions are Python 2.7.8 and Python 3.4.2.

Start with one of these versions for learning Python if you want the most stability; they're both considered stable production releases.

If you don't know which version to use, try Python 3.4.x. Some existing third-party software is not yet compatible with Python 3; if you need to use such software, you can download Python 2.7.8 instead."

8.1.2.1.2 ActiveState Website

From <http://www.activestate.com/activepython>:

"ActivePython Business and Enterprise Editions feature our precompiled, supported, quality-assured Python distribution used by millions of developers around the world for easy Python installation and quality-assured code. When you're using Python on production servers or mission-critical applications, ActivePython Business and Enterprise Editions offer significant time savings over open source Python or installing, managing, and standardizing your Python."

8.1.2.1.3 Download Python

How to check if python is already installed on your computer?

Read "<https://wiki.python.org/moin/BeginnersGuide/Download>"

For those Operating System distributions that don't include Python, various release versions can be obtained directly from the Python Software Foundation at "www.python.org".

CAUTION:

Linux and Unix Operating System distributions typically include a recent, but not necessarily the latest, Python release. However, since the Operating System distribution may itself use a specific Python release, it is often necessary to follow special installation procedures when adding newer Python releases. For example, see the "How to install Python 2.7 and Python 3.3 on CentOS 6" article at <http://toomuchdata.com>.

Microsoft Windows distributions do NOT include any Python release and do NOT include any "Curses" or "nCurses" Terminal Control Library for use with the terminal emulators (xterm, xterm-color, xterm-16color, xterm-88color, xterm-256color, vt100 and vt220) and terminals.

However, if you install Cygwin, the free Linux-like Command Line Interface and GNU Toolkit from Red Hat at "<http://cygwin.com>", you can then install a recent, but not necessarily the latest Python 2.x and/or Python 3.x release and the associated "Curses" or "nCurses" Terminal Control Library.

From Wikipedia, the free encyclopedia:

"The GnuWin32 project provides native ports in the form of runnable computer programs, patches, and source code for various GNU and open source tools and software, much of it modified to run on the 32-bit Windows platform. The ports included in the GnuWin32 packages are:

- GNU utilities such as bc, bison, chess, Coreutils, diffutils, ed, Flex, gawk, gettext, grep, Groff, gzip, iconv, less, m4, patch, readline, rx, sharutils, sed, tar, texinfo, units, Wget, which
- Archive management and compression tools, such as: arc, arj, bzip2, gzip, lha, zip, zlib.
- Non-GNU utilities such as: cygutils, file, ntfsprogs, OpenSSL, PCRE.
- Graphics tools.
- PDCurses
- Tools for processing text.
- Mathematical software and statistics Software"

NOTE:

Unfortunately, Python Curses for Microsoft Windows traps when it fails to import "_curses". Furthermore, the "tsWxGTUI" Toolkit will trap when it fails to obtain mouse button key codes from PDCurses 3.4 via these standard Python Curses module references:

- Curses.BUTTON1_RELEASED
- Curses.BUTTON1_PRESSED
- Curses.BUTTON1_CLICKED
- Curses.BUTTON1_DOUBLE_CLICKED
- Curses.BUTTON1_TRIPLE_CLICKED
- Curses.BUTTON2_RELEASED
- Curses.BUTTON2_PRESSED
- Curses.BUTTON2_CLICKED
- Curses.BUTTON2_DOUBLE_CLICKED
- Curses.BUTTON2_TRIPLE_CLICKED
- Curses.BUTTON3_RELEASED
- Curses.BUTTON3_PRESSED
- Curses.BUTTON3_CLICKED
- Curses.BUTTON3_DOUBLE_CLICKED
- Curses.BUTTON3_TRIPLE_CLICKED
- Curses.BUTTON4_RELEASED
- Curses.BUTTON4_PRESSED
- Curses.BUTTON4_CLICKED
- Curses.BUTTON4_DOUBLE_CLICKED
- Curses.BUTTON4_TRIPLE_CLICKED
- Curses.BUTTON_CTRL
- Curses.BUTTON_SHIFT
- Curses.BUTTON_ALT

8.1.2.2 Training

8.1.2.2.1 Dive Into Python

a. Official Website

From <http://www.diveintopython.net/>

"Dive Into Python is a free Python book for experienced programmers. It was originally hosted at DiveIntoPython.org, but the author has pulled down all copies. It is being mirrored here. You can read the book online, or download it in a variety of formats. It is also available in multiple languages."

b. Download Python

"ActivePython (freeware, Windows)
IDLE (open source, Windows)
Komodo (commercial, Windows)
RPMs (open source, Linux)
Source code (all platforms)
python-mode for Emacs
Jack Jansen's binaries (open source, Mac OS, Mac OS X)
Fink (open source, Mac OS X)
Jython (open source, Java)
more platforms"

8.1.2.2.2 Learn Python

"Learning to Program
How to Think Like a Computer Scientist
Thinking in Python
Introductions
O'Reilly Python Center
Non-English resources
more tutorials"

8.1.2.2.3 Python Community

"Python.org

comp.lang.python

c.l.p.announce

Tutor

Python mailing lists

more links"

8.1.2.2.4 Python Cookbook

"Python Cookbook

Vaults of Parnassus

Python.facts"

8.1.3 Release Process

8.1.3.1 Release Nomenclature

Since the TeamSTARS "tsWxGTUI" Toolkit is made available for both Python 2x and Python 3x, there must be two separate releases.

To distinguish the two releases from each other, each TeamSTARS "tsWxGTUI" Toolkit release and its associated compressed archive ("tarball" or "zip") file name is composed of three components:

- a) <Release Name>
- b) <Python Type>
- c) <Major.Minor.BugFix Number>

The components are defined as follows:

- a) Release Name:
"tsWxGTUI"
- b) Python Type:
"Py2x" for the second generation Python language syntax, semantics and global module index associated with Python 2.0.0-2.7.9
"Py3x" for the third generation Python language syntax, semantics and global module index associated with Python 3.0.0-3.4.3
- c) Major-Minor-BugFix Number:

"0.0.0" for first major, pre-alpha, release

"1.0.0" for first major, production, release

"3.2.1" for first bug fix, for second functional enhancement for third major production release

d) Examples:

"tsWxGTUI_Py2-3.2.1"

"tsWxGTUI_Py3-3.2.1"

e) Release File Type:

Linux/Unix "tarball" Type such as:

"tsWxGTUI_Py2-3.2.1.tar.gz"

"tsWxGTUI_Py3-3.2.1.tar.gz"

Windows "zip" Type such as:

"tsWxGTUI_Py2-3.2.1.zip"

"tsWxGTUI_Py3-3.2.1.zip"

8.1.3.2 Python Distribution Utilities

The TeamSTARS "tsWxGTUI" Toolkit is packaged for release using the Python Distribution Utilities ("Distutils").

Annotated Excerpt from:

<https://docs.python.org/2/distutils/introduction.html#an-introduction-to-distutils>

"Using the "Distutils" is quite simple, both for module developers and for users/administrators installing third-party modules. As a developer, your responsibilities (apart from writing solid, well-documented and well-tested code, of course!) are:

- write a setup script (setup.py by convention)
- (optional) write a setup configuration file (*MANIFEST.in*)
- create a source distribution (*tsWxGTUI_Py2x-0.0.0.tar.gz* or *tsWxGTUI_Py2x-0.0.0.zip*)
- (optional) create one or more built (binary) distributions

Each of these tasks is covered in this document.

Not all module developers have access to a multitude of platforms, so it's not always feasible to expect them to create a multitude of built distributions. It is hoped that a class of intermediaries, called packagers, will arise to address this need. Packagers will take source distributions released by module developers, build them on one or more platforms, and release the resulting built distributions. Thus, users on the most popular platforms will be able to install most popular Python module distributions in the most natural way for their platform, without having to run a single setup script or compile a line of code."

Working versions of the distribution built and install files are contained in the following directories:

"/SoftwareGadgetry-PyPI/Python-2x"

"/SoftwareGadgetry-PyPI/Python-3x"

8.1.4 Release Repository

Presuming that Toolkit release recipients will develop Toolkit application programs for subsequent release, it is suggested that recipients emulate the Toolkit Author's working software repository.

This tactic minimizes duplication of files, directories and effort. It facilitates use of file comparison and merge tools such as GNU's "Diffutils" and Deltopia's "DeltaWalker" to merge updates from the Python 2.x baseline source code into older, previously debugged Python 3.x source code thereby avoiding additional translations with Python's "2to3" utility and manual debugging of the previously fixed translation issues.

The following organization chart depicts both standard and optional features.

Key:

- [] -- Denotes a Directory containing one or more Directories and/or Files.
- "" -- Denotes Name of a Directory or File.
- + -- Denotes an organizational branch relationship.
- | -- Denotes a organizational hierarchy relationship.
- : -- Denotes an optional or temporary organizational relationship.

8.1.4.1 Release Contents

```
#-----
#"Time-stamp: <07/01/2015  8:10:46 AM rsg>
#-----

===== File: README2-Repository.txt =====

+-----+-----+ TeamSTARS "tsWxGTUI_PyVx" Toolkit
| ts | Wx |      with Python 2x & Python 3x based
+-----+-----+      Command Line Interface (CLI)
| G T U I |      and "Curses"-based "wxPython"-style,
+-----+-----+      Graphical-Text User Interface (GUI)

Get that cross-platform, pixel-mode "wxPython" feeling
on character-mode 8-/16-color (xterm-family) & non-color
(vt100-family) terminals and terminal emulators.

You can find this and other plain-text files in the
Toolkit subdirectory named:

    "./<Toolkit Recipient's Repository>/Documents".

<Your Working Repository>
(e.g. "tsWxGTUI_PyVx_Preview")
|
|   Working repository containing directories and
|   files to be packaged into downloadable "tarball"
|   and/or "zip" files via the setup shell scripts
|   at the bottom of this diagram.
|
+-- ["Documents"] (Original)
|
|   |   This directory contains a collection of files
|   |   which provide the Toolkit recipient with an
|   |   understanding of the purpose, goals & capabil-
|   |   ities, non-goals & limitations, terms & condi-
|   |   tions and procedures for installing, operating,
|   |   modifying and redistributing the Toolkit.
|   |
|   +-- "README.txt"
|   +-- "README1-Introduction.txt"
|   +-- "README2-Repository.txt"
|   +-- "README3-Documents.txt"
|   +-- "README4-ManPages.txt"
|   +-- "README5-Notebooks.txt"
|   +-- "README6-SourceDistributions.txt"
|   +-- "README7-DeveloperSandboxes.txt"
|   +-- "README8-SitePackages.txt"
|   +-- "README9-KeyBoardMouseInput.txt"
|   +-- "GETTING_STARTED.txt"
|
+-- ["ManPages"] (Original)
```

```

|
| Deliverable Toolkit manual pages are a
| form of online software documentation
| usually found on a Unix or Unix-like
| operating system.
|
| Topics covered include computer programs
| (including library and system calls),
| formal standards and conventions, and even
| abstract concepts.
|
| Unlike their Unix or Unix-like counterparts,
| a Toolkit user may NOT invoke a man page by
| issuing the "man command". Instead, a user
| must display a man page by issuing the
| "less <man document file>" command.
|
| +-- ["tsManPagesLibCLI"]
| +-- ["tsManPagesLibGUI"]
| +-- ["tsManPagesTestsLibCLI"]
| +-- ["tsManPagesTestsLibGUI"]
| +-- ["tsManPagesToolsCLI"]
| +-- ["tsManPagesToolsGUI"] (Future)
| +-- ["tsManPagesToolsLibCLI"]
| +-- ["tsManPagesToolsLibGUI"] (Future)
| +-- ["tsManPagesUtilitiesCLI"] (Future)
|
| +-- "README4-ManPages.txt"
|
+-- ["Notebooks"] (Original Pre-dates Documents)
|
| Contains a collection of commentaries that
| express opinions or offerings of explana-
| tions about events or situations that might
| be useful to Toolkit installers, developers,
| operators, troubleshooters and distributors.
| The documents may be in Application-specific
| formats (such as Adobe PDF, JPEG Bit-mapped
| image, LibreOffice, Microsoft Office, plain
| text).
|
| +-- ["DeveloperNotebook"] (Future Original
| Developer-Sandbox)
|
| Contains a collection of:
| API-References-Pixel-Mode-wxPython
| and Developer-ReadMe-Files
|
| +-- "README5-DeveloperNotebook.txt"
|
+-- ["EngineeringNotebook"] (Future Original
Developer-Sandbox)
|
| Contains a Toolkit Developer oriented collection of:
|
| Project (purpose,
| goals,

```

```

    non-goals,
    features,
    capabilities,
    limitations),

Plan (software life-cycle),

Requirements (purpose,
    goals,
    non-goals,
    features,
    capabilities,
    limitations,
    file system configuration,
    hardware & software interface,
    software,
    system,
    user configuration options),

Design (API emulation strategy, architecture),

Implementation (developer-sandbox, site-package),

Test (unit, integration, system, acceptance),

Marketing (announcement, brochure),

Release (introduction,
    release notes,
    software user's manual,
    terms & conditions,
    dictionary),

Third-party Resources

```

```

+-- "README5-EngineeringNotebook.txt"

```

```

+-- ["ProjectNotebook"] (Original Site-Package)

```

Contains a Toolkit User oriented collection of ["EngineeringNotebook"] abstracts:

```
Project (purpose,
        goals,
        non-goals,
        features,
        capabilities,
        limitations)
```

```
+-- "README5-ProjectNotebook.txt"
```

```
+-- "README5-Notebooks.txt"
```

```

+-- ["SourceDistributions"] (Original)

```

```

|   |   Contains a collection of computer program
|   |   source code files that the Toolkit recip-
|   |   ient will need to install, operate, modify
|   |   and re-distribute the Toolkit.
|
|   +-- ["Developer-Sandboxes"] (Pre-dates Site-Packages)
|   |   |
|   |   |   A sandbox is a testing environment that iso-
|   |   |   lates untested code changes and outright
|   |   |   experimentation from the production environ-
|   |   |   ment or repository.
|   |   |
|   |   +-- ["tsWxGTUI_PyVx"] (Developer-Sandbox)
|   |   |   |
|   |   |   |   +-- ["Documents"] (Copy)
|   |   |   |   |
|   |   |   |   +-- ["ManPages"] (Copy)
|   |   |   |   |
|   |   |   |   +-- ["Python-2x"] (Developer-Sandbox)
|   |   |   |   |   |
|   |   |   |   |   |   +-- ["tsWxGTUI_Py2x"]
|   |   |   |   |   |
|   |   |   |   +-- ["Python-3x"] (Developer-Sandbox,
|   |   |   |   |   |   Ported from Python-2x)
|   |   |   |   |   |
|   |   |   |   |   +-- ["tsWxGTUI_Py3x"]
|   |   |   |
|   |   +-- "README7-DeveloperSandboxes.txt"
|
|   +-- ["Site-Packages"]
|   |   |
|   |   |   Site-packages is the location where third-
|   |   |   party packages are installed (i.e., those
|   |   |   not part of the core Python distribution).
|   |   |   NOTE: That with Linux, Mac OS X and Unix
|   |   |   operating systems one must have root priv-
|   |   |   ileages to write to that location.
|   |   |
|   |   +-- ["tsWxGTUI_PyVx"] (Site-Package)
|   |   |   |
|   |   |   |   +-- ["Documents"] (Copy)
|   |   |   |   |
|   |   |   |   +-- ["ManPages"] (Copy)
|   |   |   |   |
|   |   |   |   +-- ["Python-2x"] (Site-Package)
|   |   |   |   |   |
|   |   |   |   |   |   +-- ["tsWxGTUI_Py2x"]
|   |   |   |   |   |
|   |   |   |   +-- ["Python-3x"] (Site-Package,
|   |   |   |   |   |   Ported from Python-2x)
|   |   |   |   |   |
|   |   |   |   |   +-- ["tsWxGTUI_Py3x"]
|   |   |   |
|   |   +-- "README8-SitePackages.txt"
|
|   +-- "README6-SourceDistributions.txt"

```

```

+-- "MANIFEST.in"
|
|   Deliverable File inclusion criteria list.
|
+-- "MANIFEST_template.in"
|
|   Deliverable Generic file inclusion criteria list
|   template for any Python version-specific TeamSTARS
|   "tsWxGTUI_PyVx" Toolkit.
|
+-- "MANIFEST_TREE.html"
|
|   Non-Deliverable Diagram (Multi-Level Org Chart)
|   depicting the hierarchical relationship between files
|   in the release, in Hypertext Markup Language format.
|
|   Diagram created via Command "./MANIFEST_TREE.sh".
|
+-- "MANIFEST_TREE.sh"
|
|   Deliverable POSIX-style Command Line Interface shell
|   script to generate diagrams depicting the hierarchical
|   relationship between files in the release
|   ("MANIFEST_TREE.html" and "MANIFEST_TREE.txt").
|
+-- "MANIFEST_TREE.txt"
|
|   Non-Deliverable Diagram (Multi-Level Org Chart)
|   depicting the hierarchical relationship between
|   files in the release, in Plain Text format.
|
|   Diagram created via Command "./MANIFEST_TREE.sh".
|
+-- "setup_tsWxGTUI_PyVx_Preview_tar_file.sh"
|
|   Deliverable POSIX-style Command Line Interface shell
|   script to generate downloadable "tarball" file.
|
+-- "setup_tsWxGTUI_PyVx_Preview_zip_file.sh"
|
|   Deliverable POSIX-style Command Line Interface shell
|   script to generate downloadable "zip" file.
|
+-- "README.txt"

```

===== TABLE OF CONTENTS =====

1. Repository

- 1.1 Documents
- 1.2 ManPages
- 1.3 Notebooks
- 1.4 SourceDistributions
- 1.5 Manifest

===== REPOSITORY =====

1. Repository

Excerpt From Wikipedia, the free encyclopedia:

"A software repository is a storage location from which software packages may be retrieved and installed on a computer."

This is the repository for the TeamSTARS "tsWxGTUI_PyVx" Toolkit. It is the collection of documentation and computer program source code files that is being distributed by its author in order to publish and share the intellectual property with others.

It contains the following subdirectories and files:

1.1 Documents

Contains introductory and other training information for installers, developers, operators, troubleshooters and distributors.

Toolkit software documentation is written in plain text that accompanies computer software. It either explains how it operates or how to use it, and may mean different things to people in different roles. Types of software documentation include:

- a) Requirements - Statements that identify attributes, capabilities, characteristics, or qualities of a system. This is the foundation for what shall be or has been implemented.
- b) Architecture/Design - Overview of software. Includes relations to an environment and construction principles to be used in design of software components.
- c) Technical - Documentation of code, algorithms, interfaces, and APIs.
- d) End user - Manuals for the end-user, system administrators and support staff.

1.2 ManPages

Contains a collection of man pages. A man page (short for manual page) is a form of online software documentation usually found on a Unix or Unix-like operating system. Topics covered include computer programs (including library and system calls), formal standards and conventions, and even abstract concepts.

Categories of man pages include:

- a) tsManPagesLibCLI
- b) tsManPagesLibGUI
- c) tsManPagesTestsLibCLI
- d) tsManPagesTestsLibGUI
- e) tsManPagesTestsToolsLibCLI
- f) tsManPagesTestsToolsLibGUI (Future)
- g) tsManPagesToolsCLI
- h) tsManPagesToolsGUI (Future)
- i) tsManPagesUtilities (Future)

1.3 Notebooks

Contains a collection of commentaries that express opinions or offerings of explanations about events or situations that might be useful to Toolkit installers, developers, operators, troubleshooters and distributors. The documents may be in Application-specific formats (Adobe PDF, JPEG Bit-mapped image, Microsoft Office, Plain text etc.).

The collection includes:

- a) DeveloperDocuments (API-References-Pixel-Mode-wxPython and Developer-ReadMe-Files); and
- b) EngineeringDocuments (Product Marketing Documentation, Project Documentation and Technical Documentation).

1.4 SourceDistributions

Contains a collection of computer program source code files that the Toolkit recipient will need to re-create the Toolkit.

The collection includes Developer-Sandbox and Site-Package versions of the following:

1.4.1 tsWxGTUI_Py2x

The source code files appropriate for use with Python 2.4-2.7.

1.4.2 tsWxGTUI_Py3x

The source code files appropriate for use with Python 3.0-3.4. These files are generated by converting their Python 2x counterparts with the 2to3 translation utility followed by debugging of unresolved syntax and type conversion issues.

NOTES:

- a) With Linux, Mac OS X and Unix operating systems one must have "root" or administrator privileges to write to the site-package location.
- b) If the user will not have permission to directly access the Repository but has a need to know specific contents, the system administrator should copy the appropriate contents of the Documents, ManPages and Notenook directories into each Site-Package.

The copies were not included in the distribution in order to:

- (1) avoid increasing the release development and qualification efforts; and
- (2) minimize the size of the downloadable "tar" and "zip" files.
- c) Users of third-party site-packages must explicitly import via its path from top-level package through lower-level packages to module:

```
site-package.package.module
```

- d) Examples for Python 2.4-2.7 site-packages:

```
from tsWxGTUI_Py2x.tsLibCLI import tsCxGlobals
from tsWxGTUI_Py2x.tsLibCLI import tsPlatformRunTimeEnvironment
from tsWxGTUI_Py2x.tsLibCLI import tsExceptions as tse
from tsWxGTUI_Py2x.tsLibCLI import tsLogger

from tsWxGTUI_Py2x.tsLibGUI import tsWx as wx
```

- d) Examples for Python 3.0-3.4 site-packages:

```
from tsWxGTUI_Py3x.tsLibCLI import tsCxGlobals
from tsWxGTUI_Py3x.tsLibCLI import tsPlatformRunTimeEnvironment
from tsWxGTUI_Py3x.tsLibCLI import tsExceptions as tse
from tsWxGTUI_Py3x.tsLibCLI import tsLogger

from tsWxGTUI_Py3x.tsLibGUI import tsWx as wx
```

1.5 Manifest

For a listing of the repository contents (complete with last modified date, time, size and access permissions), see:

```
"./MANIFEST_TREE.html"
"./MANIFEST_TREE.txt"
```

For additional commentary on the repository contents

see :

"./Documents/README-Manifest.txt"

===== End-Of-File =====

Draft

8.1.5 User Documentation

8.1.5.1 Equipment Operator Documents

```
#-----
#"Time-stamp: <07/01/2015  8:12:13 AM rsg>"
#-----

===== File: README3-Documents.txt =====

+-----+-----+ TeamSTARS "tsWxGTUI_PyVx" Toolkit
| ts | Wx |      with Python 2x & Python 3x based
+-----+-----+      Command Line Interface (CLI)
| G T U I |      and "Curses"-based "wxPython"-style,
+-----+-----+      Graphical-Text User Interface (GUI)

Get that cross-platform, pixel-mode "wxPython" feeling
on character-mode 8-/16-color (xterm-family) & non-color
(vt100-family) terminals and terminal emulators.

You can find this and other plain-text files in the
Toolkit subdirectory named:

    "./<Toolkit Recipient's Repository>/Documents".

<Your Working Repository>
(e.g. "tsWxGTUI_PyVx_Preview")
|
|
+-- ["Documents"]
|
|   This directory contains a collection of files
|   which provide the Toolkit recipient with an
}   understanding of the purpose, goals & capabil-
|   ities, non-goals & limitations, terms & condi-
|   tions and procedures for installing, operating,
|   modifying and redistributing the Toolkit.
|
|
+-- "AUTHORS.txt"
+-- "BUGS.txt"
+-- "CHANGE_LOG.txt"
+-- "CONFIGURE.txt"
+-- "COPYING.txt"
+-- "COPYRIGHT.txt"
+-- "CREDITS.txt"
+-- "DEMO.txt"
+-- "FAQ.txt"
+-- "GETTING_STARTED.txt"
+-- "INSTALL.txt"
+-- "LICENSE.txt"
+-- "NEWS.txt"
+-- "NOTICES.txt"
+-- "OPERATE.txt"
+-- "README.txt"
+-- "README1-Introduction.txt"
+-- "README2-Repository.txt"
```

```

|      +-- "README3-Documents.txt"
|      +-- "README4-ManPages.txt"
|      +-- "README5-Notebooks.txt"
|      +-- "README6-SourceDistributions.txt"
|      +-- "README7-DeveloperSandboxes.txt"
|      +-- "README8-SitePackages.txt"
|      +-- "README9-KeyboardMouseInput.txt"
|      +-- "THANKS.txt"
|      +-- "TO-DO.txt"
|      +-- "TROUBLESHOOT.txt"
|
+-- ["ManPages"]
|
+-- ["Notebooks"]
|
+-- ["SourceDistributions"]
|
+-- "README.txt"

===== ["DOCUMENTS"] =====

This directory contains a collection of files which provide the Toolkit recipient with an understanding of the purpose, goals & capabilities, non-goals & limitation, terms & conditions and procedures and for installing, operating, modifying and redistributing the Toolkit.

"GETTING_STARTED.txt" --- Introduces new recipients to
                           the system requirements and
                           third-party resources available
                           to new Toolkit users.

"README.txt" --- Introduces new recipients to the
                  purpose, goals, non-goals, design
                  and features of the computer
                  software product. Supplements
                  include the following:

                  "README1-Introduction.txt"
                  "README2-Repository.txt"
                  "README3-Documents.txt"
                  "README4-ManPages.txt"
                  "README5-Notebooks.txt"
                  "README6-SourceDistributions.txt"
                  "README7-DeveloperSandboxes.txt"
                  "README8-SitePackages.txt"
                  "README9-KeyboardMouseInput.txt"

"AUTHORS.txt" --- List of the principal "tsWxGTUI"
                  Toolkit author(s) and authors
                  credited for work covered by a prior
                  copyright and license.

"BUGS.txt" --- List of Known Problems / Issues.

```

"CHANGE_LOG.txt" --- List of Additions, Modification and Deletions.

"CONFIGURE.txt" --- Instructions for applying factory and site-specific configurations.

"COPYING.txt" --- Instructions for copying all or a portion of the distribution.

"FAQ.txt" --- Answers to Frequently Asked Questions.

"INSTALL.txt" --- Describes steps to download, extract install and configure the "tsWxGUI" Toolkit.

"LICENSE.txt" --- General and special arrangements, provisions, rules, specifications and standards that form an integral part of the agreement or contract between the creator and recipient of Copyrighted and Licensed Work.

"MANIFEST.txt" --- Tally List for deliverable items.

"NEWS.txt" --- Announcements of new releases.

"NOTICES.txt" --- Details the copyright(s) and license(s).

"OPERATE.txt" --- Describes steps to use the "tsWxGUI" Toolkit.

"THANKS.txt" --- Acknowledgments to those otherwise unsung heros who contributed time and effort to supporting the authors as planners, editors, designers, coders and testers.

"TO-DO.txt" --- A To-Do-List provides a roadmap for development and troubleshooting work.

"TROUBLESHOOT.txt" --- Provides a list of available reference resources and a guide for planning, developing and troubleshooting a cross-platform system of hundreds of files each containing a few, tens or hundred of class, data and method definitions. Its complexity becomes apparent in the recent software Lines-Of-Code metrics.

===== End-Of-File =====

8.1.5.2 Software Engineer and Developer Documents

```
#-----
#"Time-stamp: <07/01/2015  8:16:00 AM rsg>"
#-----

===== File: README5-Notebooks.txt =====

+-----+-----+ TeamSTARS "tsWxGTUI_PyVx" Toolkit
| ts | Wx |      with Python 2x & Python 3x based
+-----+-----+      Command Line Interface (CLI)
| G T U I |      and "Curses"-based "wxPython"-style,
+-----+-----+      Graphical-Text User Interface (GUI)

Get that cross-platform, pixel-mode "wxPython" feeling
on character-mode 8-/16-color (xterm-family) & non-color
(vt100-family) terminals and terminal emulators.

You can find this and other plain-text files in the
Toolkit subdirectory named:

    "./<Toolkit Recipient's Repository>/Documents".

<Your Working Repository>
(e.g. "Technical_Preview")
|
+-- ["Documents"]
|
+-- ["ManPages"]
|
+-- ["Notebooks"]
|
|   Contains a collection of commentaries that
|   express opinions or offerings of explana-
|   tions about events or situations that might
|   be useful to Toolkit installers, developers,
|   operators, troubleshooters and distributors.
|   The documents may be in Application-specific
|   formats (Adobe PDF, JPEG Bit-mapped image,
|   Microsoft Office, Plain text etc.).
|
|   The collection includes:
|
+-- ["DeveloperNotebook"]
|
|   +-- ["DeveloperReadMeFiles"]
|   |
|   |   Repository for entire collection of
|   |   README and topic of interest files
|   |
+-- ["API"]
|
|   Application Programming Interfaces
|
```

```

+-- ["API-tsLibCLI"]
|
|   Library of Character-mode
|   Command Line Interface Building Blocks
|
+-- ["API-tsLibGUI"]
|
|   Library of Character-mode
|   Graphical User Interface Building Blocks
|
+-- ["API-wxPython-2.8.9.2-docs"]
|
|   Library of Pixel-Mode
|   wxPython-2.8.9.2 Building Blocks
|   To-Be-Emulated
|
+-- ["API-wxPython-3.0.2.0-docs"]
|
|   Future Library of Pixel-Mode
|   wxPython-3.0.2.0 Building Blocks
|   To-Be-Emulated
|
+-- ["CLI-How-To-Files"]
|
|   Tutorial on Python programing language
|   generation evolution, differences and
|   configuration control
|
+-- "CLI_0_hello_world_print_statement.py"
+-- "CLI_1_hello_world_print_function.py"
+-- "CLI_2_hello_world_script_environment.py"
+-- "CLI_3_hello_world_main_module_application.py"
+-- "test_tsCxGlobals.py"
|
+-- ["GUI-How-To-Files"]
|
|   Tutorial on Python graphical library
|   generation evolution, differences and
|   configuration control
|
+-- "GUI_4_Curses_Widget-API-application.py"
+-- "GUI_5_tsWxGTUI_Widget-API-application.py"
+-- "GUI_6_tsWxGTUI_BoxSizer-API-application.py"
+-- "test_tsWxGlobals.py"
|
+-- "README5-DeveloperNotebook.txt")

+-- ["EngineeringNotebook"]
|
|   Contains a collection of files, in application-
|   specific formats, which provide the Toolkit archi-
|   tect, product manager, project engineer, system
|   engineer, software engineer, test engineer and
|   troubleshooter with:
|
+-- ["Adobe-PDF-Files"]
+-- ["API-Preview"]

```



```

|         |      +-- ["ASCII-Text-Files"]
|         |      +-- ["Bugzilla-Products-and-Components"]
|         |      +-- ["JPEG-Image-Files"]
|         |      +-- ["MS-Access-Files"]
|         |      +-- ["MS-Excel-Files"]
|         |      +-- ["MS-PowerPoint-Files"]
|         |      +-- ["MS-Visio-Files"]
|         |      +-- ["MS-Word-Files"]
|         |      |
|         |      +-- "README-EngineeringNotebook.txt"
|         |
|      +-- ["ProjectNotebook"]
|         |
|         |      Contains a Toolkit User oriented collection of
|         |      ["EngineeringNotebook"] abstracts:
|         |
|         |      Project (purpose,
|         |      goals,
|         |      non-goals,
|         |      features,
|         |      capabilities,
|         |      limitations)
|         |
|         |      +-- "README-ProjectNotebook.txt")
|         |
|      +-- "README5-Notebooks.txt"
|
+-- ["SourceDistributions"]
|
+-- "README.txt"

```

===== TABLE OF CONTENTS =====

1. ["Developer-Documents"]
 - 1.1 ["API-Preview"]
 - 1.2 ["API-References-Pixel-Mode-wxPython"] (Future)
 - 1.3 ["DeveloperReadMeFiles"] (Future)
2. ["Engineering-Documents"]
 - 2.1 Product Marketing Documentation (Future)
 - 2.2 Project Documentation
 - 2.3 Technical Documentation

===== ["DEVELOPER-DOCUMENTS"] =====

1. ["Developer-Documents"]

This directory contains a collection of files which provide the Toolkit Building Block, Tool and Application programmer with an understanding of the design and usage of the Toolkit's CLI and GUI Application Programming In-

terface.

1.1 ["API-References-Pixel-Mode-wxPython"]

1.2 ["DeveloperReadMeFiles"]

===== ["ENGINEERING-DOCUMENTS"] =====

2. ["Engineering-Documents"]

This directory contains a collection of files, in application-specific formats, which provide the Toolkit architect, product manager, project engineer, system engineer, software engineer, test engineer and troubleshooter with:

2.1 Product Marketing Documentation

- a) announcement notice(s)
- b) brochure(s)
- c) introduction

2.2 Project Documentation

- a) goal and capability objectives
- b) non-goal and limitation constraints
- c) project plans

2.3 Technical Documentation

- a) architectural plans
- b) system specifications
- c) Interface specifications
- d) software specifications
- e) design specifications
- f) test specifications
- g) software user manual

===== End-Of-File =====

8.2 Installation Procedure

8.2.1 Developer Platform Procedure (Rough Draft)

- 1 Download developer version of "tsWxGTUI" Toolkit compressed "tarball" file
 - a) Copy compressed "tarball" (SoftwareGadgetry-Dev-20130608-095615.tgz) from TBD to temporary folder /myDownloads/tsWxGTUI
- 2 Extract "tsWxGTUI" Toolkit components from compressed "tarball" file
 - a) cd /myDownloads/tsWxGTUI
 - b) mkdir myWR/
 - c) gunzip SoftwareGadgetry-Dev-20130608-095615.tgz

- d) `cd myWR`
- e) `tar xvf ../SoftwareGadgetry-Dev-20130608-095615.tar`
- 3** Execute run time environment build and install procedure that adds "tsWxGTUI" Toolkit to Python site-packages.
 - a) `cd myDownloads/tsWxGTUI/myWR/SoftwareGadgetry-Dev/Python-2.x`
 - b) `python setup.py build`
 - c) `python setup.py install`
- 4** Verify installation by running set of test applications.
 - a) `python test_tsOperatorSettingsParser.py`
 - b) `python test_tsWxWidgets.py`

8.2.2 Operator Platform Procedure

- 1** Download operator version "tsWxGTUI" Toolkit compressed "tarball" file
 - a) Copy compressed "tarball" (SoftwareGadgetry-User-20130608-095615.tgz) from TBD to temporary folder /myDownloads/tsWxGTUI
- 2** Extract "tsWxGTUI" Toolkit components from compressed "tarball" file
 - a) `cd /myDownloads/tsWxGTUI`
 - b) `mkdir myWR/`
 - c) `gunzip SoftwareGadgetry-User-20130608-095615.tgz`
 - d) `cd myWR`
 - e) `tar xvf ../SoftwareGadgetry-User-20130608-095615.tar`
- 3** Execute run time environment build and install procedure that adds "tsWxGTUI" Toolkit to Python site-packages.
 - a) `cd myDownloads/tsWxGTUI/myWR/SoftwareGadgetry-User/Python-2.x`
 - b) `python setup.py build`
 - c) `python setup.py install`
- 4** Verify installation by running set of test applications.
 - a) `python test_tsOperatorSettingsParser.py`
 - b) `python test_tsWxWidgets.py`

8.3 User Interface Design

8.3.1 Command Line Interface

- 1 Command Line Syntax <application> <keyword options> <positional arguments>.
- 2 Command Line Examples:

Command Line Example	Description
ls	ls program gets a brief list of information about the FILES (in the current directory by default).
ls --help	Keyword option (--help) get a description of the usage of the ls program and its keyword options and positional arguments
ls -la	Keyword options -la gets an annotated list of information about the FILES (the current directory by default).
python tsLinesOfCode.py	python interpreter program default version uses its positional argument tsLinesOfCode.py to specify the name of the python command line tool to be executed
python2.7 tsLinesOfCode.py	python interpreter program version 2.7 uses its positional argument tsLinesOfCode.py to specify the name of the python command line tool to be executed. The tool uses default values for all of its keyword options
python3.3 tsLinesOfCode.py -i /WR/SoftwareGadgetry-Dev/Python-3.x -o tsWxGTUI_Developer_Metrics.txt	python interpreter program version 3.3 uses its positional argument tsLinesOfCode.py to specify the name of the python command line tool to be executed. The tool uses the keyword-value pair option i to specify the source code input directory to be scanned and the -o option to specify the name of the output report file. It uses default values for any unspecified keyword options

8.3.2 Graphical User Interface

REGION	DESCRIPTION	FEATURES
Top Of Desktop	Application Windows (Frames and Dialogs) are constrained to occupy all columns of the designated Screen Client Area which is directly above Redirected Stdio (Frame).	<p>Top Level Windows (Frames and Dialogs) are constrained to be located within Screen Border at the appropriate character cell coordinates (i.e., spaced every 8 pixels horizontally and 12 pixels vertically). Centering is subject to the same Screen Border and character cell coordinate constraints.</p> <ul style="list-style-type: none"> ▪ Border is an optional line surrounding the Client Area of the GUI object ▪ Top Border Title is optional text identifying the GUI object to the operator ▪ Top Border Buttons are optional controls available to the operator for Frames and Dialogs ▪ Client Area Menu Bar are optional activities available to the operator ▪ Client Area Tool Bar are optional utilities available to the operator ▪ Client Area Buttons, CheckBoxes and Radio Boxes & Buttons are application controls available to the operator ▪ Client Area Status Bar are application activity and progress indicators available to the operator. The Status Bar may be subdivided into two or more panels (fields) that are constrained to be located within the parent Frame border. The wx.ThemeToUse provides the means to better utilize narrow screens (by overlapping Status Bar panel borders) and to indicate hidden text (by the use of ellipses (i.e., "...")).
Middle of Desktop	Redirected Stdio (Frame) is constrained to occupy all columns of the designated rows immediately above the Task Bar.	<ul style="list-style-type: none"> ▪ Border is a line surrounding the Client Area of the GUI object. ▪ Top Border Title identifies the Redirected Stdio GUI object. ▪ Client Area displays the most recent event messages of interest to the operator. The messages are date and time stamped. The messages are labeled with the severity level associated with the event and any information describes the issue.

Bottom Of Desktop	Task Bar (Frame) is constrained to occupy all columns of the designated bottom rows of the screen.	<ul style="list-style-type: none"> ▪ Border is a line surrounding the Client Area of the GUI object ▪ Top Border Title identifies the Task Bar GUI object and the runtime name of the application. ▪ Client Area Buttons are GUI object controls available to the operator for changing which GUI object reappears after having been iconized and which appears in front of all others. ▪ Client Area Status Bar are application activity and progress indicators available to the operator. The normally twirling button field updates when the application is idle or non-twirling when busy or stuck. The date and time field allows the operator to assess the timeliness of the display.
-------------------	--	---

8.4 Developer

Draft

9 PERFORMANCE TUNING

- 1** Layout of the display begins with the start of the GUI application. It is revised as the GUI application introduces new features.
- 2** Output of the display features begins only after the GUI application has initiated the first Show. Output has had only minimal optimization.
- 3** Ideally the Task Bar clock should update every second, however the update interval typically alternates between 1 and 2 seconds suggesting that the busy time average is nominally 1.5 seconds. (Note: During development, the system incurs a substantial, but unknown amount of overhead due to its frequent logging of various diagnostic activities.

Draft

Draft

10 ACCEPTANCE TESTING

None.

Draft

Draft

11 COMPONENT STATUS (as of 2013/12/02)

Source File	Limitations	Extensions
tsApplication	Usable.	
tsCommandLineInterface	Usable.	
tsConfig	Usable.	
tsCxGlobals	Usable.	
tsDecorators	Usable.	
tsExceptions	Usable.	
tsLinesOfCode	Usable.	
tsLogger	Usable.	
tsOptionParser	Usable.	
tsPlatformQuery	Usable.	
tsPlatformRunTimeEnvironment	Usable.	
tsPublish	Usable.	
tsReportUtilities	Usable.	
tsSysCommands	Usable.	
tsThreadPool	Usable.	
tsTreeCopy	Usable.	
tsTreeTrimLines	Usable.	
tsTreeTrimShutil	Usable.	
tsWx	Usable. Configuration file for application developer use. Not all wxPython constants are included. Constants for colors include all the expected wxPython ones via a mapping to one of the the available curses color values (black, white, red, magenta, blue, cyan, green and yellow).	Configuration file provides the means for a developer to tailor the look and feel of the wxPython emulation.
tsWxAcceleratorEntry	Usable.	
tsWxAcceleratorTable	Usable.	

Source File	Limitations	Extensions
tsWxApp	Usable. The main section of each applications must be wrapped in a new control structure that introduces tsAppplication, tsExceptions and tsLogger modules.	The tsAppplication, tsExceptions and tsLogger modules enable applications to be integrated with others for automatic operation. Acting as any "well behaved UNIX application" the curses based GUI can now return an exit code to the command line shell and display an associated exception statement and back trace.
tsWxBoxSizer	Usable.	
tsWxButton	Usable. Limited real estate precludes use of top and bottom borders. Left and right borders are simulated by use of square brackets "[" and "]".	
tsWxCallLater		
tsWxCaret	Usable.	
tsWxCheckBox	Usable.	
tsWxChoice		
tsWxCliLinesOfCode	Usable.	
tsWxColor	Usable.	
tsWxColorDataBase	Usable.	
tsWxCommandLineEnv	Usable.	
tsWxControl	Usable.	None.
tsWxControlWithItems	Usable.	
tsWxCursor	Usable.	
tsWxDebugHandlers		
tsWxDialog	Usable.	
tsWxDisplay	Usable.	Constructor accepts an optional flag dictionary for passing reserved space for task bar and stdio redirection. Methods for getting dimensions of screen geometry, screen client area, task bar, and stdio redirection area accept optional argument to request character rather than pixel dimensions.
tsWxDoubleLinkedList	Usable.	
ysWxEraseEvent		
tsWxEvent	Usable.	

Source File	Limitations	Extensions
tsWxEventDaemon		
tsWxEventLoop	Usable.	
tsWxEventLoopActivator	Usable.	
tsWxEventLoopEntry		
tsWxEvtHandler	Usable.	
tsWxFlexGridSizer	Under Construction.	
tsWxFocusEvent	Under Construction.	
tsWxFrame	Usable. Position and size are constrained to fit within the portion of the display screen allocated by the command line shell. This module does not yet support event driven resizing by the operator.	Frame layout algorithm supports task bar, stdio redirection and application type frames. It coerces application frames to fit within the display's client area.
tsWxGauge	Usable.	
tsWxGlobals	Usable. Configuration file for use only by Software Engineers responsible for organization, application and toolkit look and feel. Not all wxPython constants are included. Constants for colors include all the expected wxPython ones via a mapping to one of the the available curses color values (black, white, red, magenta, blue, cyan, green and yellow).	Configuration file provides the means for a developer to tailor the look and feel of the wxPython emulation.
tsWxGraphicalTextUser Interface	Usable. The underlying Python curses module does NOT yet support the deletion of GUI objects. It does not support Microsoft Window platforms without the Cygwin plug-in. The Cygwin plug-in supports mouse events only in xterm sessions.	
tsWxGridBagSizer	Under Construction.	
tsWxGridSizer	Usable.	
tsWxItemContainer	Under Construction.	

Source File	Limitations	Extensions
tsWxKeyEvent	Under Construction.	
tsWxLayoutAlgorithm	Under Construction. Instantiation raises the unimplemented exception.	
tsWxLinesOfCodeProjectMetrics	Usable.	
tsWxListBox		
tsWxLog		
tsWxMenu	Under Construction.	
tsWxMenuBar	Under Construction. Supports only the building of top level menu selections. It does not yet support event driven activities or the building of lower level menus.	
tsWxMouseEvent	Usable.	
tsWxMouseState	Usable.	
tsWxMultiFrameEnv	Usable.	
tsWxObject	Usable.	None.
tsWxPanel	Usable.	
tsWxPasswordEntryDialog	Usable. Use curses.textpad which is NOT event driven	
tsWxPoint	Usable.	None.
tsWxPyApp	Usable. Minimal support for such application methods as OnPreInit, OnInit etc.	
tsWxPyEventBinder	Usable.	
tsWxPyOnDemandOutputWindow	Usable.	Configurable via tsWxGlobals' ThemeToUse. Options include Foreground/Background color, Title, Timestamp and Window Height.
tsWxPySimpleApp	Usable.	None.
tsWxPySizer	Usable.	
tsWxRadioBox	Usable.	
tsWxRadioButton	Usable.	
tsWxRect	Usable.	None.
tsWxScreen	Usable.	
tsWxScrollBar	Usable.	
tsWxScrollBarButton	Usable.	

Source File	Limitations	Extensions
tsWxScrollBarGauge	Usable.	
tsWxScrolled	Usable.	
tsWxScrolledText	Usable.	
tsWxScrolledWindow	Usable.	
tsWxShowEvent		
tsWxSize	Usable.	None.
tsWxSizer	Usable.	
tsWxSizerFlags	Usable.	
tsWxSizerItem	Usable.	
tsWxSizerItemList	Usable.	
tsWxSizerSpacer	Usable.	
tsWxSlider	Under Construction.	
tsWxSplashScreen	Usable. However, bit-mapped image must be created and saved before use. The bit-mapped image must be loaded by tsWxGraphicalTextUserInterface before the initialization of any tsWx module.	
tsWxStaticBox	Usable.	
tsWxStaticBoxSizer	Usable.	
tsWxStaticText	Usable. May NOT yet support text updates.	
tsWxStatusBar	Usable. May NOT yet support text updates. Does NOT support icons or other non-text objects. Border of the the parent fram and the status bar (and its associated sub-panels) are overlapped to conserve display screen real estate.	Configurable via tsWxGlobals' ThemeToUse. Options include Ellipses for truncated text, Border Margins, and Window Height.
tsWxSystemSettings	Usable.	
tsWxTaskBar	Usable (Display only).	
tsWxTextCtrl	Usable. Supports line and word wrap.	Configurable via tsWxGlobals' ThemeToUse. Report Utility Options include page width, page length and

Source File	Limitations	Extensions
		page layout.
tsWxTextEditBox	Under Development.	This is an extension to wxPython.
tsWxTextEntryDialog	Usable. However, it is event driven and blinks annoyingly.	
tsWxTimer		
tsWxToggleButton	Under Construction.	
tsWxTopLevelWindow	Usable..	None.
tsWxValidator	Under Construction.	
tsWxWindow	Usable.	

12 APPENDIXES

13 APPENDIX A - BASELINE HOST COMPUTER PLATFORMS

This section shall identify those host computer platforms actually used to plan, implement, test, document, deploy and maintain the "tsWxGTUI_PyVx" Toolkit.

- 1** *Currently Used Platforms* (see "*Currently Used Platforms (Release Notes)*" on page 57) - Linux (Fedora 22 / OpenSUSE 13.2 / Scientific (CentOS) 7 / Ubuntu 12.04, 14.04 & 15.04), Mac OS X (10.7 / 10.11), Microsoft Windows (7 Pro. 32-bit with Cygwin 1.7.x / 10 Pro. 64-bit with Cygwin 2.2) / and Unix (PC-BSD 10 & OpenIndiana 151a8 / Solaris 11 / SunOS 5.11).

The Toolkit's Python source code has been developed and tested only with Intel x86 and x64 processors and representative GNU/Linux, Mac OS X, Microsoft Windows and Unix operating system releases.

Its source code has been compiled, interpreted and executed by Python 2x and 3x Virtual Machines developed by the Python Software Foundation (PSF).

The PSF also distributes equivalent Python 2x and 3x Virtual Machines (and the source code to build them) for other processor types and operating systems, some of which are listed below.

See file `"./tsWxGTUI_PyVx_Repository/Notebooks/EngineeringNotebook/MS-Excel-Files/Platform_Configuration.xls"` for Python version specific platform configuration details (Sheet 1 --- Host Configurations; Sheet 2 --- Terminal Configurations):

- a) 2013-model year, 3.5 GHz Intel Quad Core i7 processor-based desktop with 16 GB RAM and sufficient performance, resources and expansion capabilities to serve as the high-level baseline development and operator platform.
 - b) 2007-model year, 2.33 GHz Intel Core 2 Duo processor-based laptop with 4 GB RAM and sufficient performance, resources and expansion capabilities to serve as the moderate-level baseline development and operator platform.
 - c) 1998-model year, 366 MHz Intel Pentium II-based laptop with 384 MB RAM and expansion capabilities (marginal resources and performance) sufficient enough to serve as the low-level baseline development and operator platform.
- 2** *Previously Used Platforms* (see "*Previously Used Platforms (Release Notes)*" on page 62) - Linux (Fedora 15 & 16 / Open SuSE 11 / Ubuntu 10.04), Mac OS X (10.4 / 10.5 / 10.6 / 10.7), Microsoft Windows (8 Professional / 7 Professional / XP Professional each with Cygwin 1.7.x) and Unix (OpenIndiana 151a6 / Solaris 11 / SunOS 5.11).
- a) 2007-model year, 2.33 GHz Intel Core 2 Duo processor-based laptop with 4 GB RAM and sufficient performance, resources and expansion capabilities to serve as the moderate-level baseline development and operator platform.

- b) 1998-model year, 366 MHz Intel Pentium II-based laptop with 384 MB RAM and expansion capabilities (marginal resources and performance) sufficient enough to serve as the low-level baseline development and operator platform.
- 3** Designing Embedded Systems with Linux and Python - Link to YouTube Video by Mark Kohler.

13.1 Currently Used Platforms (Release Notes)

Host Computer Platform Configurations currently used by "tsWxGTUI_PyVx" Toolkit developers:

Draft

Draft

Make & Model	Hardware	Software
Apple 27" iMac Desktop	<p>2013-model year, 3.5 GHz Intel Quad Core i7 processor-based desktop with 16 GB RAM, 2560x1440 pixel resolution display and sufficient performance, resources and expansion capabilities to serve as the high-level baseline development and operator platform.</p> <ul style="list-style-type: none"> Its 3 TB (7200 RPM) SATA 6 Gb/s internal hard drive had 128 GB Solid State Flash memory and was used to store and run Apple's Mac OS X and the hypervisor virtualization applications (Parallels Desktop for Mac and VMware Fusion for Mac) that supported various guest operating systems. Hewlett-Packard Company P2015DN Series (26A5C8) LaserJet Printer connected via Ethernet Hewlett-Packard Company OfficeJet Pro 8500A (A910) e-All-in-One Series Printer connected via Wi-Fi or USB. 	<p>Host Operating System</p> <ul style="list-style-type: none"> Apple's Mac OS X (initially Mavericks releases 10.9.x and currently Sierra releases 10.12.x) with Python 2.7.10, Python 3.5.0 and Python 3.6.0 <p>Host Hypervisor Virtualization Applications</p> <ul style="list-style-type: none"> Parallels Desktop 9-12 for Mac VMware Fusion 5 & 7.1.3 for Mac <p>Concurrent or Interchangeable Guest Operating Systems (cloned from Apple 17" MacBook Pro Laptop and configured to share host drives and use 1440 x 900 display)</p> <ul style="list-style-type: none"> eComStation Demo CD version 2.2 BETA (IBM OS/2 4.5 Warp descendant) runs only what was shipped on the live CD and does not include Python. Most recent Python port for OS/2 & eComStations is Python 2.7.5. For details, see the following: "http://os2ports.smedley.id.au/index.php?page=python" and "http://blog.python.org/2011/05/python-33-to-drop-support-for-os2.html" GNU/Linux (CentOS 7.2 64-bit, Debian 8.5.0 64-bit, Fedora 22-24 64-bit, LXLE 14.02 32-bit, OpenSUSE 12.2 & 13.2 64-bit, Scientific 6.4 & 7.2 64-bit, Ubuntu 12.04, 14.04, 15.05 & 16.04 32-/64-bit with Python 2.7.x and 3.4.x/3.5.x. Microsoft Windows (10.0 Professional 64-bit, 10.0 Professional 32-bit, 8.1 Professional 32-bit, 8 Professional 32-bit, 7 Professional 32-bit with SP1, XP Professional 32-bit with SP3 and 2000 Professional 32-bit with SP2) with POSIX-like Cygwin plug-in and Python 2.6, 2.7, 3.2, 3.3 and 3.4. (NOTE: Windows 2000 came with obsolete Web Browser that precluded Windows 2000 updates and installation of Cygwin. After copying Windows 2000 updates and Cygwin directory from XP, Windows 2000 ran the TeamSTARS "tsWxGTUI_PyVx" Toolkit CLI and GUI tests but did not support mouse even with xterm.) UNIX (PC-BSD 9.2 & 10.3 64-bit without Parallels Tools, OpenIndiana 151a8 32-bit without Parallels Tools, a replacement for unreleased OpenSolaris 11/SunOS 5.11)

		with Python 2.6.4 running on Apple MacBook Pro
Apple 17" MacBook Pro Laptop	<p>2007-model year, 2.33 GHz Intel Core 2 Duo processor-based laptop with 4 GB RAM, 1920x1200 pixel resolution display and sufficient performance, resources and expansion capabilities to serve as the moderate-level baseline development and operator platform.</p> <ul style="list-style-type: none"> Its 160 GB (5400 RPM) SATA 1.5 Gb/s internal hard drive was used to run Apple's Mac OS X 10.7 and the hypervisor virtualization applications (Parallels Desktop 8 and VMware Fusion 5) that supported various guest operating systems. Its 1.5 TB (7200 RPM) SATA 3 Gb/s external hard drive was used to store and run configured versions of the Ubuntu Linux (12.04), Microsoft Windows (8 Pro, 7 Pro and XP Pro) and Unix (OpenSolaris 11/OpenIndiana 151) guest operating systems. Hewlett-Packard Company P2015DN Series (26A5C8) LaserJet Printer connected via Ethernet Hewlett-Packard Company OfficeJet Pro 8500A (A910) e-All-in-One Series Printer connected via Wi-Fi or USB. 	<p>Host Operating System</p> <ul style="list-style-type: none"> Apple's Mac OS X 10.7.5 with Python 2.6.8, 2.7.5, 3.2.2 <p>Host Hypervisor Virtualization Applications</p> <ul style="list-style-type: none"> Parallels Desktop 8 VMware Fusion 5 <p>Interchangeable Guest Operating Systems (configured to share host drives and use 1440 x 900 display)</p> <ul style="list-style-type: none"> Linux (Fedora 20 64-bit, OpenSUSE 12.2 64-bit, Scientific (CentOS) 6.4-6.5 64-bit, Ubuntu 12.04 32-bit) with Python 2.7 and 3.2. Windows (8.1 Professional 32-bit, 8 Professional 32-bit, 7 Professional 32-bit with SP1, XP Professional 32-bit with SP3) with POSIX-like Cygwin plug-in and Python 2.6, 2.7, 3.2 and 3.3 UNIX (PC-BSD 9.2-10.0 64-bit without Parallels Tools, OpenIndiana 151A8 32-bit without Parallels Tools, a replacement for unreleased OpenSolaris 11/SunOS 5.11) with Python 2.6.4 running on Apple MacBook Pro
Dell 15.6" Inspiron 7000 Laptop	<p>1998-model year, 366 MHz Intel Pentium II-based laptop with 384 MB RAM, 640x480/1024x768 pixel resolution display and expansion capabilities (marginal resources and performance) sufficient enough to serve as the low-level baseline development and operator platform.</p> <ul style="list-style-type: none"> Its interchangeable 32 GB (4200 RPM) ATA hard drives were used to run Windows XP Pro or Ubuntu 12.04 Linux. The platform's limited memory and available PCMCIA network adapters were incompatible with later versions of Windows or with other Linux distributions. (Windows XP recognized Xircom Ethernet and 3Com Wireless adapters; Ubuntu Linux 12.04 recognized only Linksys Wireless adapter.) Hewlett-Packard Company Photosmart C3180 All-in-One Printer, Scanner, Copier connected via USB. 	<p>Interchangeable Host Operating System</p> <ul style="list-style-type: none"> Windows XP Professional with SP3) with POSIX-like Cygwin plug-in and Python 2.6, 2.7, 3.2 and 3.3 Linux (Ubuntu 12.04) with Python 2.7 and 3.3

NOTE: Since cross-platform operating system and Python virtual machine technology is also available for

non-Intel based systems, it is likely that the TeamSTARS "tsWxGTUI_PyVx" toolkit will also work on those systems which use the equivalent operating systems and Python virtual machines with 32-bit and 64-bit microprocessors from other manufacturers including:

- *AMD*
- *ARM Holdings*
- *Cyrix*
- *Freescape*
- *Intel*
- *IBM*
- *Marvell*
- *NexGen*
- *Nvidia Tegra*
- *Oracle (previously Sun)*
- *OWC*
- *Qualcomm*
- *Rise Technology*
- *Samsung*
- *SigmaTel*
- *Texas Instruments*
- *Transmeta*
- *tilera*
- *Via (Centaur Technology division)*
- *winchip*

13.2 Previously Used Platforms (Release Notes)

Host Computer Platform Configurations previously used by "tsWxGTUI_PyVx" Toolkit developers:

Make & Model	Hardware	Software
Apple 17" MacBook Pro Laptop	<p>2007-model year, 2.33 GHz Intel Core 2 Duo processor-based laptop with 4 GB RAM and sufficient performance, resources and expansion capabilities to serve as the moderate-level baseline development and operator platform.</p> <ul style="list-style-type: none"> Its 160 GB (5400 RPM) SATA 1.5 Gb/s internal hard drive was used to run Apple's Mac OS X 10.7 and the hypervisor virtualization applications (Parallels Desktop 4-7 and VMware Fusion 4-5) that supported various guest operating systems. Its 1.5 TB (7200 RPM) SATA 3 Gb/s external hard drive was used to store and run configured versions of the Ubuntu Linux (10.04), Microsoft Windows (7 Pro and XP Pro) and Unix (OpenSolaris 11/OpenIndiana 151) guest operating systems. 	<p>Host Operating System</p> <ul style="list-style-type: none"> Apple's Mac OS X 10.4-10.7 with Python 2.6.8 <p>Host Hypervisor Virtualization Applications</p> <ul style="list-style-type: none"> Parallels Desktop 4-7 VMware Fusion 4-5 <p>Interchangeable Guest Operating Systems (configured to share host drives and use 1440 x 900 display)</p> <ul style="list-style-type: none"> Linux (Ubuntu 10.04) with Python 2.7 and 3.2 Windows (7 Professional / XP Professional with SP3) with POSIX-like Cygwin plug-in and Python 2.6, 2.7 and 3.2 UNIX (OpenIndiana 151a5, a replacement for unreleased OpenSolaris 11/SunOS 5.11) with Python 2.6.4 running on Apple MacBook Pro
Dell 15.6" Inspiron 7000 Laptop	<p>1998-model year, 366 MHz Intel Pentium II-based laptop with 384 MB RAM and expansion capabilities (marginal resources and performance) sufficient enough to serve as the low-level baseline development and operator platform.</p> <ul style="list-style-type: none"> Its interchangeable 32 GB (4200 RPM) ATA hard drives were used to run Windows XP Pro or Ubuntu 12.04 Linux. The platform's limited memory and available PCMCIA network adapters were incompatible with later versions of Windows or with other Linux distributions. (Windows XP recognized Xircom Ethernet and 3Com Wireless adapters; Ubuntu Linux 12.04 recognized only Linksys Wireless adapter.) 	<p>Interchangeable Host Operating System</p> <ul style="list-style-type: none"> Windows XP Professional with SP3) with POSIX-like Cygwin plug-in and Python 2.6, 2.7 and 3.2 Linux (Ubuntu 12.04) with Python 2.7 and 3.2

Notes - Baseline Host Computer Platform Configurations previously used by "tsWxGTUI_PyVx" Toolkit developers	<ol style="list-style-type: none">1 Linux (Fedora 15-16) with Python 2.6, 2.7 and 3.2 running on Apple MacBook Pro - 32-/64-bit Intel Core 2 Duo Processors running with 4 GB RAM Linux Virtual Machine created and managed by Parallels Desktop 6 for Mac2 Linux (OpenSUSE 11) with Python 2.6, 2.7 and 3.2 running on Apple MacBook Pro - 32-/64-bit Intel Core 2 Duo Processors with 4 GB RAM running Virtual Machine created and managed by Parallels Desktop 6 for Mac3 Linux (Ubuntu 10.04) with Python 2.7 and 3.2 running on Apple MacBook Pro - 32-/64-bit Intel Core 2 Duo Processors with 4 GB RAM running Linux Virtual Machine created and managed by Parallels Desktop 6 for Mac4 Mac OS X (Tiger 10.4.0-Snow Leopard 10.6.8) with Python 2.6, 2.7 and 3.2 running on Apple MacBook Pro - 32-/64-bit Intel Core 2 Duo Processors with 4 GB RAM running Mac OS 10.4-10.6.5 Windows (XP-SP3) with UNIX-like Cygwin plug-in and Python 2.6 running on Dell Laptop - 32-bit Intel Pentium 2 Processor with 384 MB RAM running Windows XP-SP36 Windows (XP-SP3 and Release 8 Preview) with UNIX-like Cygwin plug-in and Python 2.6 running on Apple MacBook Pro - 32-/64-bit Intel Core 2 Duo Processors with 4 GB RAM running Windows Virtual Machine created and managed by Parallels Desktop 6 for Mac
---	---

Notes - Experimental Host Computer Platform Configurations previously used by "tsWxGTUI_PyVx" Toolkit developers	<p>1 Windows (7 Professional) with built-in "Command Prompt" accessory shell and Python 2.7 running on Apple MacBook Pro - 32-/64-bit Intel Core 2 Duo Processors with 4 GB RAM running Windows Virtual Machine created and managed by Parallels Desktop (8) for Mac.</p> <hr/> <p>This configuration uses the Windows built-in "Command Prompt" accessory shell which can run Command Line Interface components ("tsLibCLI", "tsTestsCLI" and "tsToolsCLI") of the "tsWxGTUI_PyVx" toolkit.</p> <p>It does NOT support the low-level, "nCurses" library, a pre-requisite for running the Graphical-style User Interface components ("tsLibGUI", "tsTestsGUI" and "tsToolsGUI") of the "tsWxGTUI_PyVx" toolkit.</p>
	<p>2 Windows (7 Professional) with UNIX-like Git Bash plug-in and Python 2.7 running on Apple MacBook Pro - 32-/64-bit Intel Core 2 Duo Processors with 4 GB RAM running Windows Virtual Machine created and managed by Parallels Desktop (8) for Mac.</p> <hr/> <p>This configuration, like those using Cygwin, includes a Bash shell which can run Command Line Interface components ("tsLibCLI", "tsTestsCLI" and "tsToolsCLI") of the "tsWxGTUI_PyVx" toolkit.</p> <p>Unlike those configurations using Cygwin, it does NOT support the low-level, "nCurses" library, a pre-requisite for running the Graphical-style User Interface components ("tsLibGUI", "tsTestsGUI" and "tsToolsGUI") of the "tsWxGTUI_PyVx" toolkit.</p>
	<p>3 Windows (7 Professional) and Python 2.7 with the GNUwin32 and PDCurses Version 2.6 plug-ins running on Apple MacBook Pro - 32-/64-bit Intel Core 2 Duo Processors with 4 GB RAM running Windows Virtual Machine created and managed by Parallels Desktop (8) for Mac.</p> <hr/> <p>This configuration, unlike those using Cygwin, includes a DOS-like Command Prompt shell which can run Command Line Interface components ("tsLibCLI", "tsTestsCLI" and "tsToolsCLI") of the "tsWxGTUI_PyVx" toolkit.</p> <p>Like those configurations using Cygwin, PDCurses Version 2.6 does support the low-level, Python "Curses" library, a pre-requisite for running the Graphical-style User Interface components ("tsLibGUI", "tsTestsGUI" and "tsToolsGUI") of the "tsWxGTUI_PyVx" toolkit. It runs the test_PDCurses (renamed test_tsWxCurses) application. However, PDCurses Version 2.6 traps because it lacks the mouse button definitions needed by the "tsWxGraphicalTextUserInterface" module to emulate "wxPython" in order to run such applications as "test_tsWxWidgets.py".</p>

Draft

14 APPENDIX B - API RELATIONSHIP

An Application Programming Interface (API) is a source code based specification intended to be used as an interface by software components to communicate with each other. An API may include specifications for routines, data structures, object classes, and variables.

- 1 High-level API** - A programming interface that is the least detailed but provides more functionality within one statement than a lower-level interface. High-level interfaces are designed to enable the programmer to write code in a shorter amount of time and to be less involved with the details of the software module or hardware device that is performing the required services. For example, the programmer can invoke a high-level procedure to append one or more text strings to an internal buffer.
- 2 Low-level API** - A programming interface that is the most detailed, allowing the programmer to manipulate functions within a software module or within hardware at a very granular level. To continue with the afore mentioned example, a high-level procedure can then invoke one or more low-level procedures that will sequentially get one text string at a time from the internal buffer, sequentially scroll the top most string off the display, then scroll up each remaining displayed string and finally outputting the new string to the bottom row of the screen.
- 3 Extended API** - A customized version of an existing programming interface that supports additional keyword value pairs and positional arguments. For example, the "tsWxGTUI_PyVx" Toolkit is a character-mode emulation of the pixel-mode "wxPython" API. It internally may require optional parameters to distinguish such features as character-mode dimensions from their pixel-mode counterparts. It may also include functionality that "wxPython" and its underlying "wxWidgets" toolkit otherwise obtain from the host operating system and its programming libraries, such as the installation of the color palette and the implementation of scrollable windows.

14.1 Comparison with "wxPython"

This tabulation shall briefly compare the features, capabilities and limitations of the pixel-mode "wxPython" / "wxWidgets" / "wxEmbedded" Toolkit with those of its character-mode emulator, the TeamSTARS "tsWxGTUI_PyVx" Toolkit.

Feature	"wxPython" / "wxWidgets" / "wxEmbedded"	TeamSTARS "tsWxGTUI_PyVx" Toolkit
Platform	Locally (via system console or xterm) and remotely (via vnc) accessible systems: <ul style="list-style-type: none"> ▪ Desktop ▪ Laptop ▪ Workstation ▪ Embedded (via KOAN's "wxEmbedded") 	Locally (via system console or xterm) and remotely (via xterm) accessible systems: <ul style="list-style-type: none"> ▪ Desktop ▪ Laptop ▪ Workstation ▪ Embedded
Operator Desktop	<ul style="list-style-type: none"> ▪ Keyboard 	<ul style="list-style-type: none"> ▪ Keyboard

Feature	"wxPython" / "wxWidgets" / "wxEmbedded"	TeamSTARS "tsWxGTUI_PyVx" Toolkit
Platform	<p>Locally (via system console or xterm) and remotely (via vnc) accessible systems:</p> <ul style="list-style-type: none"> ▪ Desktop ▪ Laptop ▪ Workstation ▪ Embedded (via KOAN's "wxEmbedded") 	<p>Locally (via system console or xterm) and remotely (via xterm) accessible systems:</p> <ul style="list-style-type: none"> ▪ Desktop ▪ Laptop ▪ Workstation ▪ Embedded
Interface Hardware	<ul style="list-style-type: none"> ▪ Mouse, trackball, touchpad or touchscreen ▪ Optional Mouseless ▪ Display (68-color pixel mode) 	<ul style="list-style-type: none"> ▪ Mouse, trackball, touchpad or touchscreen ▪ Keyboard Shortcut Key and/or Optional Mouse (Future) ▪ Display (68-color character mode mapped into: 8-color/64-color pair on cygwin, linux, xterm or xterm-color terminals) ▪ Display (71-color character mode mapped into: 16-color/256-color pair on xterm-16color, xterm-88color or xterm-256color terminals) ▪ Display (a single shade of white, green or orange that is "ON" depending on the single available phosphor or black when "OFF" on vt100/vt220 terminals)
Operating System & Device Driver Software	<ul style="list-style-type: none"> ▪ GNU/Linux (Linux Operating System Kernel with Unix-like GNU Command Line Interface, Graphical User Interface and Toolchain). ▪ Mac OS X (Darwin-/BSD-based Unix Operating System Kernel with GNU Command Line Interface, Toolchain and Apple Computer's Graphical User Interface and Toolchain). ▪ Microsoft Windows (with free add-on of POSIX-like Cygwin Command Line Interface and GNU tools from Red Hat) ▪ Unix (Operating System Kernel with Unix-like Command Line Interface, Graphical User Interface and Toolchain) 	<ul style="list-style-type: none"> ▪ GNU/Linux (Linux Operating System Kernel with Unix-like GNU Command Line Interface, Graphical User Interface and Toolchain). ▪ Mac OS X (Darwin-/BSD-based Unix Operating System Kernel with GNU Command Line Interface, Toolchain and Apple Computer's Graphical User Interface and Toolchain). ▪ Microsoft Windows (with free add-on of POSIX-like Cygwin Command Line Interface and GNU tools from Red Hat) ▪ Unix (Operating System Kernel with Unix-like Command Line Interface, Graphical User Interface and Toolchain)
Terminal Device Interface Software	<ul style="list-style-type: none"> ▪ "wxWidgets" internally uses host Operating System specific API for its Terminal Device Interface. ▪ It translates host Operating System specific events into "wxWidgets" specific published API events. ▪ Events include keyboard key press / release, mouse button press / release and single / double click with mouse position at every clocked sample. ▪ Events also include window resizing. 	<ul style="list-style-type: none"> ▪ "tsWxGTUI_PyVx" internally uses Python "Curses" module specific API for its Terminal Device Interface. ▪ Python "Curses" module internally uses host Operating System specific "Curses" and/or "nCurses" API for its Terminal Device Interface. ▪ It translates "Curses" module specific events into published "wxWidgets" specific API events. ▪ Events include keyboard key press / release,

Feature	"wxPython" / "wxWidgets" / "wxEmbedded"	TeamSTARS "tsWxGTUI_PyVx" Toolkit
Platform	<p>Locally (via system console or xterm) and remotely (via vnc) accessible systems:</p> <ul style="list-style-type: none"> ▪ Desktop ▪ Laptop ▪ Workstation ▪ Embedded (via KOAN's "wxEmbedded") 	<p>Locally (via system console or xterm) and remotely (via xterm) accessible systems:</p> <ul style="list-style-type: none"> ▪ Desktop ▪ Laptop ▪ Workstation ▪ Embedded
		<p>mouse button press / release and single / double / triple click with mouse position ONLY available at time of button state change.</p> <ul style="list-style-type: none"> ▪ Events also include window resizing. However, event handling is currently limited to trapping the unsupported event. Though re-initializing "Curses" to detect and use the new size is feasible and fast (50 milliseconds or more depending on host processor and terminal color palette (and associated definition of or mapping of wxPython color palette), it does not address the time consuming (20 seconds or more on host processor) complexities associated with re-initializing the "wxPython" emulation and the System Operator designated application program.
Programming Language	<ul style="list-style-type: none"> ▪ "wxWidgets" is implemented in C++ ▪ "wxPython" binding/wrapper for "wxWidgets" is implemented with SWIG in Python 2.x and Python 3.x 	<ul style="list-style-type: none"> ▪ Primary baseline "tsWxGTUI_Py2x" is implemented in Python 2.x and then debugged. ▪ Secondary baseline "tsWxGTUI_Py3x" is implemented in Python 3.x as a "port" from "tsWxGTUI_Py2x" via the standard Python "2to3" translation utility followed by minor manual debugging when appropriate
Terminal Interface Software	<ul style="list-style-type: none"> ▪ "wxWidgets" internally uses Operating System or X window system specific API for Graphical User Interface. 	<ul style="list-style-type: none"> ▪ "tsWxGTUI_PyVx" internally uses Python Curses module API for Graphical-style User Interface. ▪ The Python Curses module only implements the most commonly used subset features of the host "Curses" or "nCurses" API. ▪ The "tsWxGTUI_PyVx" Toolkit emulates a typical Operating System or X window system specific API for Graphical User Interface. It establishes the appropriate relationship between a triggering event (mouse button click) and the GUI object (button, gauge, frame, dialog, panel etc.) to receive and subsequently handle the triggering event notification.

Feature	"wxPython" / "wxWidgets" / "wxEmbedded"	TeamSTARS "tsWxGTUI_PyVx" Toolkit
Platform	<p>Locally (via system console or xterm) and remotely (via vnc) accessible systems:</p> <ul style="list-style-type: none"> ▪ Desktop ▪ Laptop ▪ Workstation ▪ Embedded (via KOAN's "wxEmbedded") 	<p>Locally (via system console or xterm) and remotely (via xterm) accessible systems:</p> <ul style="list-style-type: none"> ▪ Desktop ▪ Laptop ▪ Workstation ▪ Embedded
Operator Desktop Interface Software	<ul style="list-style-type: none"> ▪ Host Operating System specific Command Line Interface is a POSIX compatible shell (such as bash) window process ▪ Host Operating System specific Graphical User Interface features support multiple independently re-sizable and repositionable Frame and Dialog processes ▪ Host Operating System specific Graphical User Interface features reflect proprietary or industry standard placement of labels and buttons to iconize, maximize/restore and close frames and dialogs ▪ Host Operating System specific Graphical User Interface task bar features support the closing of individual Frames and Dialogs ▪ Host Operating System specific task bar features support the shifting of foreground focus from one Frame or Dialog to another 	<ul style="list-style-type: none"> ▪ Host Operating System specific Command Line Interface is a POSIX compatible shell (such as bash) window process ▪ Host Operating System specific Graphical User Interface utilizes the existing Command Line Interface shell window process ▪ "tsWxGTUI_PyVx" emulated Graphical User Interface features DO NOT support multiple independently re-sizable and repositionable Frames and Dialogs WITHOUT the System Operator first creating separate Command Line Interface shell window processes ▪ "tsWxGTUI_PyVx" emulated Graphical User Interface features reflect Microsoft Windows-style placement of labels and buttons to iconize, maximize/restore and close frames and dialogs ▪ "tsWxGTUI_PyVx" emulated Graphical User Interface task bar features DO NOT currently support the closing of individual Frames and Dialogs ▪ "tsWxGTUI_PyVx" Toolkit task bar features (future enhancement) support the shifting of foreground focus from one Frame or Dialog to another
Application Programming Interface	<ul style="list-style-type: none"> ▪ "wxWidgets" / "wxPython" API for Graphical User Interface supports bit-mapped images. ▪ It supports fixed and variable sized fonts and at least the 68 most commonly used colors. 	<ul style="list-style-type: none"> ▪ "tsWxGTUI_PyVx" emulated subset of "wxWidgets" / "wxPython" API for Graphical-style User Interface DOES NOT support bitmapped images except for the single, predefined bitmapped image used as a splash screen at startup ▪ It supports a single fixed sized font and at least the 68 most commonly used colors (which are internally mapped into the "Curses" standard 8-color or 16-color terminal hardware and terminal emulator software). Future "Curses" enhancements may enable support a single fixed sized font and at least the 68 most commonly used colors defined for optional 88-color and 256-color terminal hardware and terminal

Feature	"wxPython" / "wxWidgets" / "wxEmbedded"	TeamSTARS "tsWxGTUI_PyVx" Toolkit
Platform	<p>Locally (via system console or xterm) and remotely (via vnc) accessible systems:</p> <ul style="list-style-type: none"> ▪ Desktop ▪ Laptop ▪ Workstation ▪ Embedded (via KOAN's "wxEmbedded") 	<p>Locally (via system console or xterm) and remotely (via xterm) accessible systems:</p> <ul style="list-style-type: none"> ▪ Desktop ▪ Laptop ▪ Workstation ▪ Embedded
		<p>emulator software.</p> <ul style="list-style-type: none"> ▪ It supports the 1-color (white, green or orange depending on the single available phosphor) that is "ON" or "OFF" (black) associated with a VT100/VT220 Terminal emulator.

Draft

14.2 High, Low and Extended API Relationships

The following tables describes the conceptual purpose, scope, capabilities, limitations and relationship between the High-level, Low-Level and Extended APIs associated with the "tsWxGTUI_PyVx" Toolkit and its third-party components.

Low-level GUI "Curses"-style API Classes and associated Class Methods:

- **Pads** - A pad is like a window, except that it is not restricted by the screen size, and is not necessarily associated with a particular part of the screen. Pads can be used when a large window is needed, and only a part of the window will be on the screen at one time. Automatic refreshes of pads (such as from scrolling or echoing of input) do not occur. The refresh() and noutrefresh() methods of a pad require 6 arguments to specify the part of the pad to be displayed and the location on the screen to be used for the display. The arguments are pminrow, pmincol, sminrow, smincol, smaxrow, smaxcol; the p arguments refer to the upper left corner of the pad region to be displayed and the s arguments define a clipping box on the screen within which the pad region is to be displayed.
- **Panels** - Panels are windows with the added feature of depth, so they can be stacked on top of each other, and only the visible portions of each window will be displayed. Panels can be added, moved up or down in the stack, and removed.
- **Windows** - a window is a rectangular area of the display with or without a border
- **Text** - A character string of one or more mono-spaced alpha-numeric, punctuation or line-draw characters.
- **Colors** - Support for terminals and terminal emulators (cygwin, linux, xterm, xterm-color, xterm-16color, xterm-88color or xterm-256color) with mouse and Red-Green-Blue color phosphors who's individual intensities can be adjusted to create a palette of terminal / terminal emulator dependent colors, ranging from 8 to 256 used to create foreground/background color pair combinations ranging from 8 x 8 to 256 x 256. However, currently the maximum number of color pairs is limited to 16 x 16.
- **Non-color** - Support for Digital Equipment Corporation VT100 and VT220 terminals and terminal emulators with/without mouse having only a single color phosphor (such as green orange or white) who's intensity can be either ON or OFF.

High-level API	Low-level API	Extended API
<p>Local Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		<p>Local & Remote Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell echo \$TERM # Display default type \$TERM=<xterm-family member ID> or <vt100-family member ID> echo \$TERM # Display changed type launch local application terminate local application login to remote platform host via "ssh <login ID>@<Host ID>:" launch remote application terminate remote application logout of remote platform host transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" terminate local shell logout of local platform host
	<p>Local Host Operating System Process API</p> <ul style="list-style-type: none"> Process Launcher Process Terminator Process Cleanup Process Event Handlers Process Input/Output Handlers 	<p>Local & Remote Host Operating System Process API</p> <ul style="list-style-type: none"> Process Launcher Process Terminator Process Cleanup Process Event Handlers Process Input/Output Handlers
		<p>Command Line Interface Low-level API (ts)</p> <ul style="list-style-type: none"> tsApplication (Class and run time library component initializes and configures the application using the following keyword-value pairs and positional arguments: input provided on the command line by an operator and input provided in the parameter list of the application's invocation of the class instantiation.) tsCommandLineEnv (Class and run time library component provides platform independent configuration,

High-level API	Low-level API	Extended API
<p>Local Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		<p>Local & Remote Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell echo \$TERM # Display default type \$TERM=<xterm-family member ID> or <vt100-family member ID> echo \$TERM # Display changed type launch local application terminate local application login to remote platform host via "ssh <login ID>@<Host ID>:" launch remote application terminate remote application logout of remote platform host transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" terminate local shell logout of local platform host
		<p>initialization, input/output supervisor and application launcher, event handler and terminator)</p> <ul style="list-style-type: none"> tsCommandLineInterface (Class and methods that prompt or re-prompt the operator for input, validate that the operator has supplied the expected number of inputs and that each is of the expected type.) tsCxGlobals (Module to establish configuration constants and macro-type functions for the Command Line Interface mode of the "tsWxGTUI_PyVx" Toolkit.) tsDoubleLinkedList (Class to establish a representation of a linked list with forward and backward pointers.) tsExceptions (Class to define and handle error exceptions. Maps run time exception types into 8-bit exit codes and prints associated diagnostic message and traceback info.) tsLogger (Class that emulates a subset of Python logging API. It defines and

High-level API	Low-level API	Extended API
<p>Local Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		<p>Local & Remote Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell echo \$TERM # Display default type \$TERM=<xterm-family member ID> or <vt100-family member ID> echo \$TERM # Display changed type launch local application terminate local application login to remote platform host via "ssh <login ID>@<Host ID>:" launch remote application terminate remote application logout of remote platform host transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" terminate local shell logout of local platform host
		<p>handles prioritized, time and date stamped event message formatting and output to files and devices. Files are organized in a date and time stamped directory named for the launched application. Unix-type devices include syslog, stderr, stdout and stdscr (the ncurses display screen). It also supports "wxPython"-style logging of assert and check case results.)</p> <ul style="list-style-type: none"> tsOperatorSettingsParser (Class to parse the command line entered by the operator of an application program. Parsing extracts the Keyword-Value pair Options and Positional Arguments that will configure and control the application during its execution.) tsPlatformRunTimeEnvironment (Class to capture current hardware, software and network information about the run time environment for the user process.) tsReportUtilities (Class defining

High-level API	Low-level API	Extended API
<p>Local Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		<p>Local & Remote Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell echo \$TERM # Display default type \$TERM=<xterm-family member ID> or <vt100-family member ID> echo \$TERM # Display changed type launch local application terminate local application login to remote platform host via "ssh <login ID>@<Host ID>:" launch remote application terminate remote application logout of remote platform host transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" terminate local shell logout of local platform host
		<p>methods used to format information: date and time (begin, end and elapsed), file size (with kilo-, mega-, giga-, tera-, peta-, exa-, zeta- and yotta-byte units) and nested Python dictionaries.)</p> <ul style="list-style-type: none"> tsSysCommands (Class definition and methods for issuing shell commands to and receiving responses from the host operating system.)
<p>Graphical User Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host and its GUI Desktop launch local application from GUI Desktop terminate local application from GUI Desktop logout of local platform host and its GUI Desktop 		<p>Local & Remote Graphical User Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell echo \$TERM # Display default type \$TERM=<xterm-family member ID> or <vt100-family member ID> echo \$TERM # Display changed type launch local application terminate local application login to remote platform host via "ssh <login ID>@<Host ID>:" launch remote application

High-level API	Low-level API	Extended API
<p>Local Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		<p>Local & Remote Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell echo \$TERM # Display default type \$TERM=<xterm-family member ID> or <vt100-family member ID> echo \$TERM # Display changed type launch local application terminate local application login to remote platform host via "ssh <login ID>@<Host ID>:" launch remote application terminate remote application logout of remote platform host transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" terminate local shell logout of local platform host
		<ul style="list-style-type: none"> terminate remote application logout of remote platform host transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" terminate local shell logout of local platform host
	<p>Local Host Operating System Process API</p> <ul style="list-style-type: none"> Process Launcher Process Terminator Process Cleanup Process Event Handlers Process Input/Output Handlers 	<p>Local & Remote Host Operating System Process API</p> <ul style="list-style-type: none"> Process Launcher Process Terminator Process Cleanup Process Event Handlers Process Input/Output Handlers
<p>Graphical User Interface Application Launcher API (wxWidgets & wxPython)</p> <ul style="list-style-type: none"> run time library components provide platform specific configuration, initialization, input/output supervisor and 	<p>Graphical User Interface Application Launcher API (nCurses)</p> <ul style="list-style-type: none"> application run time library components provide platform specific nCurses configuration, initialization, input/output supervisor and application 	<p>Graphical User Interface Application Launcher API (tsWxGTUI)</p> <ul style="list-style-type: none"> tsWx (Module to load all symbols that should appear within the wxPython.wx emulation namespace. Included are various classes, constants, functions and methods

High-level API	Low-level API	Extended API
<p>Local Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		<p>Local & Remote Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell echo \$TERM # Display default type \$TERM=<xterm-family member ID> or <vt100-family member ID> echo \$TERM # Display changed type launch local application terminate local application login to remote platform host via "ssh <login ID>@<Host ID>:" launch remote application terminate remote application logout of remote platform host transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" terminate local shell logout of local platform host
application launcher	launcher	<p>available for use by applications built with components from the wxPython emulation infrastructure.)</p> <ul style="list-style-type: none"> tsWxGlobals (Module to establish configuration constants and macro-type functions for the Graphical-style User Interface mode of the "tsWxGTUI_PyVx" Toolkit. Provides a centralized mechanism for modifying/restoring those configuration constants that can be interrogated at runtime by those software components having a "need-to-know". The intent being to avoid subsequent searches to locate and modify or restore a constant appropriate to the current configuration. Provides a theme-based mechanism for modifying / restoring those configuration constants.) tsWxMultiFrameEnv (Class to enable an application using a Command Line Interface (CLI) to launch and use the same character-mode terminal with a Graphical-style User Interface (GUI). It uses application specified

High-level API	Low-level API	Extended API
<p>Local Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		<p>Local & Remote Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell echo \$TERM # Display default type \$TERM=<xterm-family member ID> or <vt100-family member ID> echo \$TERM # Display changed type launch local application terminate local application login to remote platform host via "ssh <login ID>@<Host ID>:" launch remote application terminate remote application logout of remote platform host transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" terminate local shell logout of local platform host
		<p>configuration keyword-value pair options to initialize any application specific logger(s) It wraps the CLI, underlying the GUI, and the GUI with exception handlers to control the exit codes and messages used to coordinate other application programs.)</p>
<p>Graphical User Interface High-level Application Launcher API (wxWidgets & wxPython)</p> <ul style="list-style-type: none"> PyApp (start wxPython application) PySimpleApp (start simple wxPython application) PyOnDemandOutput (start Redirected Output window) 		<p>Graphical User Interface High-level Application Launcher API (tsWxGTUI)</p> <ul style="list-style-type: none"> PyApp (start wxPython application) PySimpleApp (start simple wxPython application) PyOnDemandOutput (start Redirected Output window; enhancements include date, time and message severity level annotations with and without font style and foreground/background color markup attributes)

High-level API	Low-level API	Extended API
<p>Local Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		<p>Local & Remote Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell echo \$TERM # Display default type \$TERM=<xterm-family member ID> or <vt100-family member ID> echo \$TERM # Display changed type launch local application terminate local application login to remote platform host via "ssh <login ID>@<Host ID>:" launch remote application terminate remote application logout of remote platform host transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" terminate local shell logout of local platform host
	<p>Graphical User Interface Low-level Launcher (nCurses)</p> <ul style="list-style-type: none"> start (curses.start) stop (curses.stop) 	<p>Graphical User Interface Low-level Launcher API (tsWxGTUI)</p> <ul style="list-style-type: none"> start (tsWxGTUI.start) stop (tsWxGTUI.stop) tsWxGraphicalTextUserInterface (Class uses the Standard Python Curses API to initialize, manage and shutdown input, from a keyboard and mouse, and output, to a two-dimensional display screen. It identifies user terminal make, model and features. It controls terminal device startup, shutdown and exception handling. It translates "wxPython"-style terminal color, pixel and character parameters into their "Curses" counterparts. Upon startup, it briefly restores or creates, saves or loads the splash screen bitmap image as specified in the "tsWxGlobals" configuration file.)
<p>GUI (wxWidgets & wxPython)</p> <ul style="list-style-type: none"> It is a C++ library that lets developers create applications for Windows, OS X, Linux 	<p>GUI (nCurses)</p> <ul style="list-style-type: none"> It is a programming library that provides an API which allows the programmer to write text mode 	<p>GUI (tsWxGTUI)</p> <ul style="list-style-type: none"> It is a Python and nCurses based programming library that lets developers create wxWidgets and

High-level API	Low-level API	Extended API
<p>Local Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		<p>Local & Remote Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell echo \$TERM # Display default type \$TERM=<xterm-family member ID> or <vt100-family member ID> echo \$TERM # Display changed type launch local application terminate local application login to remote platform host via "ssh <login ID>@<Host ID>:" launch remote application terminate remote application logout of remote platform host transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" terminate local shell logout of local platform host
<p>and UNIX on 32-bit and 64-bit architectures as well as several mobile platforms including Windows Mobile, iPhone SDK and embedded GTK+.</p>	<p>user interfaces in a terminal independent manner.</p> <ul style="list-style-type: none"> It is a toolkit for developing "GUI-like" application software that runs under a terminal emulator. It also optimizes screen changes, in order to reduce the latency experienced when using remote shells. 	<p>wxPython style applications for Windows, OS X, Linux and UNIX on 32-bit and 64-bit architectures.</p> <ul style="list-style-type: none"> It emulates a subset of the wxWidgets and wxPython API that is suitable for text mode terminals. It requires the operator to pre-adjust the position, size and appearance of each command shell window, within the display, before running an application program.
<ul style="list-style-type: none"> It has popular language bindings for Python, Perl, Ruby and many other languages. 		
<ul style="list-style-type: none"> Unlike other cross-platform toolkits, wxWidgets gives its applications a truly native look and feel because it uses the platform's native API rather than emulating the GUI. 		
<ul style="list-style-type: none"> It's also extensive, free, open-source and mature. 		

High-level API	Low-level API	Extended API
<p>Local Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		<p>Local & Remote Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell echo \$TERM # Display default type \$TERM=<xterm-family member ID> or <vt100-family member ID> echo \$TERM # Display changed type launch local application terminate local application login to remote platform host via "ssh <login ID>@<Host ID>:" launch remote application terminate remote application logout of remote platform host transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" terminate local shell logout of local platform host
<ul style="list-style-type: none"> It supports local terminals with various keyboard, mouse and pixel mode display device configurations. 	<ul style="list-style-type: none"> It supports local and remote terminals with various keyboard, mouse and text mode display device configurations. 	<ul style="list-style-type: none"> It supports local and remote terminals with various keyboard, mouse and text mode display device configurations.
<ul style="list-style-type: none"> It enables the operator to adjust the display position, size and appearance of the top level GUI objects. 	<ul style="list-style-type: none"> It requires the operator to pre-adjust the position, size and appearance of each command shell window, within the display, before running an application program. It requires the operator to login to each remote computer before running an application program. 	<ul style="list-style-type: none"> It requires the operator to login to each remote computer before running an application program.
<ul style="list-style-type: none"> It enables application programmers to control the initial display position, size and appearance of the top level GUI objects. 	<ul style="list-style-type: none"> It enables application programmers to control the initial position, size and appearance of the top level GUI objects, within a command shell window. Application programs may or may not malfunction or terminate after an operator re-adjusts the size of the command shell window. For example, the Midnight Commander application from the Free 	<ul style="list-style-type: none"> It enables application programmers to control the initial position, size and appearance of the top level GUI objects, within a command shell window. Application programs will malfunction or terminate after an operator re-adjusts the size of the command shell window.

High-level API	Low-level API	Extended API
<p>Local Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		<p>Local & Remote Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell echo \$TERM # Display default type \$TERM=<xterm-family member ID> or <vt100-family member ID> echo \$TERM # Display changed type launch local application terminate local application login to remote platform host via "ssh <login ID>@<Host ID>:" launch remote application terminate remote application logout of remote platform host transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" terminate local shell logout of local platform host
	<p>Software Foundation automatically adjusts itself to changes in screen size when used with an xterm type terminal emulator but needs to be manually refreshed via Ctrl-L when used with a cygwin console shell.</p>	
<p>Host Platform High-level Interface (Host)</p> <p>Platform-specific API, libraries and User Interface for starting, initializing, configuring, controlling, monitoring and terminating computer hardware and software activities.</p> <ul style="list-style-type: none"> Linux (Fedora 17-20 & Ubuntu) 10.04-12.04 Mac OS X (Lion 10.7.4-10.9.1) Microsoft Windows (XP with SP3, 7, 8 & 8.1) with Cygwin (1.7.2) add-on 	<p>Host Platform Low-level Interface (Virtual Machine)</p> <p>Programming language specific API, platform-specific libraries and User Interface for starting, initializing, configuring, controlling, monitoring and terminating application software activities.</p> <ul style="list-style-type: none"> Python 2.6.x, 2.7.x and/or Python 3.x (provides platform independent Virtual Machine API) Python Curses Module (provides terminal independent keyboard, mouse and display API) NCurses Library (implements 	<p>Host Platform Low-level Interface (tsWxGTUI analogous to host services provided by "gtk", "msw", "osx" and "unix")</p> <p>Programming language specific API, platform-specific libraries and User Interface for starting, initializing, configuring, controlling, monitoring and terminating application software activities.</p> <ul style="list-style-type: none"> tsWxGraphicalTextUserInterface (Class uses the Standard Python Curses API to initialize, manage and shutdown input, from a keyboard and mouse, and output, to a two-dimensional display screen. It identifies user terminal make, model and features. It controls terminal

High-level API	Low-level API	Extended API
<p>Local Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		<p>Local & Remote Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell echo \$TERM # Display default type \$TERM=<xterm-family member ID> or <vt100-family member ID> echo \$TERM # Display changed type launch local application terminate local application login to remote platform host via "ssh <login ID>@<Host ID>:" launch remote application terminate remote application logout of remote platform host transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" terminate local shell logout of local platform host
<ul style="list-style-type: none"> Unix (SunOS 5.11) 	<p>platform specific keyboard, mouse and display API)</p>	<p>device startup, shutdown and exception handling. It translates "wxPython"-style terminal color, pixel and character parameters into their "Curses" counterparts. Upon startup, it briefly restores or creates, saves or loads the splash screen bitmap image as specified in the "tsWxGlobals" configuration file.)</p>
<ul style="list-style-type: none"> 	<p>Top Level Windows (wxWidgets & wxPython)</p> <ul style="list-style-type: none"> Frame (A GUI object whose size and position can usually be changed by the user. It usually has thick borders and a title bar, and can optionally contain a menu bar, toolbar and status bar. A frame can contain any GUI object that is not a frame or dialog. Frames are windows associated with an application task (such as an internet WEB Browser) that can be minimized into an icon, expanded to full screen or terminated upon operator demand.) 	<p>Top Level Windows ((tsWxGTUI implements feature(s) available in wxWidgets or wxPython library)</p> <ul style="list-style-type: none"> Frame (A GUI object whose size and position can (usually) be changed by the user. It usually has thick borders and a title bar, and can optionally contain a menu bar, toolbar and status bar. A frame can contain any GUI object that is not a frame or dialog. Frames are windows associated with an application task (such as an internet WEB Browser) that can be minimized into an icon, expanded to full screen or terminated upon operator demand.) Dialog (A GUI object, such as PasswordEntryDialog or

High-level API	Low-level API	Extended API
<p>Local Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		<p>Local & Remote Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell echo \$TERM # Display default type \$TERM=<xterm-family member ID> or <vt100-family member ID> echo \$TERM # Display changed type launch local application terminate local application login to remote platform host via "ssh <login ID>@<Host ID>:" launch remote application terminate remote application logout of remote platform host transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" terminate local shell logout of local platform host
	<ul style="list-style-type: none"> Dialog (A GUI object, such as PasswordEntryDialog or TextEntryDialog, with a title bar and sometimes a system menu, which can be moved around the screen. It can contain controls and other GUI objects and is often used to allow the user to make some choice or to answer a question. Dialogs are pop-up windows associated with an application task's menu bar (such as an input field for an operator's search criteria and an output field for a list of candidate WEB sites) that will be terminated upon completion.) Redirected Output (An optional window to display the UNIX-style "stdout" (progress) and "stderr" (error) messages output by an application task via a "print" statement.) 	<p>TextEntryDialog, with a title bar and sometimes a system menu, which can be moved around the screen. It can contain controls and other GUI objects and is often used to allow the user to make some choice or to answer a question. Dialogs are pop-up windows associated with an application task's menu bar (such as an input field for an operator's search criteria and an output field for a list of candidate WEB sites) that will be terminated upon completion.)</p> <ul style="list-style-type: none"> Redirected Output (An optional window to display the UNIX-style "stdout" (progress) and "stderr" (error) messages output by an application task via a "print" statement. All output is automatically prefixed with the date (year/month/day); the time (hour:minute:second.millisecond such as "2011/01/23 12:34:56.789"; a separator (" - "); an optional severity level indicator (DEBUG, WARNING, ERROR etc. When the application designer wants to draw attention to

High-level API	Low-level API	Extended API
<p>Local Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		<p>Local & Remote Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell echo \$TERM # Display default type \$TERM=<xterm-family member ID> or <vt100-family member ID> echo \$TERM # Display changed type launch local application terminate local application login to remote platform host via "ssh <login ID>@<Host ID>:" launch remote application terminate remote application logout of remote platform host transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" terminate local shell logout of local platform host
		<p>special messages, the messages may be enhanced by: Reversed or custom foreground and background colors; blink, bold, dim, normal, standout and underline special effects)</p>
		<p>Top Level Windows (tsWxGTUI implements feature(s) not available in wxWidgets or wxPython library)</p> <ul style="list-style-type: none"> Redirected Output (AppendText applies Color and Font Attribute Markup) TaskBar (buttons shift focus in manner similar to native host GUI that underlies wxWidgets and wxPython)

High-level API	Low-level API	Extended API
<p>Local Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		<p>Local & Remote Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell echo \$TERM # Display default type \$TERM=<xterm-family member ID> or <vt100-family member ID> echo \$TERM # Display changed type launch local application terminate local application login to remote platform host via "ssh <login ID>@<Host ID>:" launch remote application terminate remote application logout of remote platform host transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" terminate local shell logout of local platform host
	<p>Low-level Windows (wxWidgets & wxPython)</p> <ul style="list-style-type: none"> Buttons (one or more action initiating selection) Check Boxes (one or more Left & Right Aligned independent feature selecting buttons) Gauges (Horizontal & Vertical bar graphs) Menu Bars (row of one or more pull down menu selections) Menus (column of one or more action initiating selections) 	<p>Low-level Windows ((tsWxGTUI implements feature(s) available in wxWidgets or wxPython library)</p> <ul style="list-style-type: none"> Buttons (one or more action initiating selection) Check Boxes (one or more Left & Right Aligned independent feature selecting buttons) Gauges (Horizontal & Vertical bar graphs) Menu Bars (row of one or more pull down menu selections) Menus (column of one or more action initiating selections)

High-level API	Low-level API	Extended API
<p>Local Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		<p>Local & Remote Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell echo \$TERM # Display default type \$TERM=<xterm-family member ID> or <vt100-family member ID> echo \$TERM # Display changed type launch local application terminate local application login to remote platform host via "ssh <login ID>@<Host ID>:" launch remote application terminate remote application logout of remote platform host transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" terminate local shell logout of local platform host
	<ul style="list-style-type: none"> Panels (optionally labeled window containing other GUI objects) Radio Boxes (Horizontal & Vertical panel containing two or more radio buttons) Radio Buttons (two or more Left & Right Aligned mutually exclusive feature selecting buttons) ScrollBar (panel with Horizontal and/or Vertical buttons, that control a scrollable text window, and a gauge, to display the position and size of the displayed text relative to the non-displayed text) ScrolledWindow (panel with a Horizontal and/or Vertical ScrollBar and a scrollable text window) 	<ul style="list-style-type: none"> Panels (optionally labeled window containing other GUI objects) Radio Boxes (Horizontal & Vertical panel containing two or more radio buttons) Radio Buttons (two or more Left & Right Aligned mutually exclusive feature selecting buttons) ScrollBar (panel with Horizontal and/or Vertical buttons, that control a scrollable text window, and a gauge, to display the position and size of the displayed text relative to the non-displayed text) ScrolledWindow (panel with a Horizontal and/or Vertical ScrollBar and a scrollable text window)

High-level API	Low-level API	Extended API
<p>Local Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		<p>Local & Remote Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell echo \$TERM # Display default type \$TERM=<xterm-family member ID> or <vt100-family member ID> echo \$TERM # Display changed type launch local application terminate local application login to remote platform host via "ssh <login ID>@<Host ID>:" launch remote application terminate remote application logout of remote platform host transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" terminate local shell logout of local platform host
	<ul style="list-style-type: none"> Splash Screen (Optional window to display a bitmap description of the application. It briefly appears before any top-level wxPython-style application Frames.) StaticText (panel for displaying text) Status Bars (panel with one or more Status Bar Panels) Status Bar Panels (panel for displaying a single piece of status information) TextCtrl (panel for displaying text that may be optionally formatted and scrolled) Tool Bars (row of one or more action initiating selections) 	<ul style="list-style-type: none"> Splash Screen (Optional window to display a text-based, bitmap-like description of the application. It briefly appears before any top-level wxPython-style application Frames.) StaticText (panel for displaying text) Status Bars (panel with one or more Status Bar Panels) Status Bar Panels (panel for displaying a single piece of status information) TextCtrl (panel for displaying text that may be optionally formatted and scrolled) Tool Bars (row of one or more action initiating selections)

High-level API	Low-level API	Extended API
<p>Local Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		<p>Local & Remote Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell echo \$TERM # Display default type \$TERM=<xterm-family member ID> or <vt100-family member ID> echo \$TERM # Display changed type launch local application terminate local application login to remote platform host via "ssh <login ID>@<Host ID>:" launch remote application terminate remote application logout of remote platform host transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" terminate local shell logout of local platform host
		<p>Low-level Windows ((tsWxGTUI implements feature(s) not available in wxWidgets or wxPython library)</p> <ul style="list-style-type: none"> Buttons (BORDER_SIMPLE for single line label replaced by bracketed label) Panels (optionally labeled window containing other GUI objects) ScrollBar Buttons (implements feature(s) not available in nCurses library) ScrollBar Gauges (implements feature(s) not available in NCurses library) Scrolled (Template for scrollable text windows with Horizontal and/or Vertical ScrollBars) Scrolled Text (implements text markup feature(s) not available in NCurses library) ScrolledWindow (scrollable text window with Horizontal and/or Vertical ScrollBars)Task Bar Buttons (implements feature(s) not available in NCurses library to apply keyboard focus shift) TextCtrl (AppendText applies Color and Font Attribute Markup)

High-level API	Low-level API	Extended API
<p>Local Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		<p>Local & Remote Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell echo \$TERM # Display default type \$TERM=<xterm-family member ID> or <vt100-family member ID> echo \$TERM # Display changed type launch local application terminate local application login to remote platform host via "ssh <login ID>@<Host ID>:" launch remote application terminate remote application logout of remote platform host transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" terminate local shell logout of local platform host
<p>GUI Object Styles (wxWidgets & wxPython)</p> <ul style="list-style-type: none"> Overlapping Tiled Border None Border Simple Splash Screen (Bitmap Image) 	<p>GUI Object Layout Sizers (wxWidgets & wxPython)</p> <ul style="list-style-type: none"> Box (automatically scaled Horizontal & Vertical Layout) Grid (automatically scaled Horizontal & Vertical Layout) Splash Screen (Bitmap Image display) Static Box (combines Box Sizer with Bordered Panel layout) 	<p>GUI Object Styles (tsWxGTUI)</p> <ul style="list-style-type: none"> Box (automatically scaled Horizontal & Vertical Layout) Grid (automatically scaled Horizontal & Vertical Layout) Splash Screen (off-line, NCurses-style Bitmap Image builder utility) Static Box (combines Box Sizer with Bordered Panel layout)
<p>Text (wxWidgets & wxPython)</p> <ul style="list-style-type: none"> Multiple Proportional (such as Times and Helvetica) and Monospaced (Courier) Font Families Multiple Sizes (8pt to 288pt) Special Effects (numerous including horizontal, vertical, rotated etc.) 	<p>Text (nCurses)</p> <ul style="list-style-type: none"> Single Monospaced (configurable by terminal operator such as Courier) Font Single Size (configurable by terminal operator such as 12pt having 8 pixel width and 12 pixel height) Special Effects (configurable by application programmer but limited to one or a combination of the following: Blinking, Bold, Dim, Normal, Reverse, Standout) 	<p>Text (tsWxGTUI)</p> <ul style="list-style-type: none"> Single Monospaced Font Single Size (conversions between wxWidgets pixel and nCurses character dimensions presumes 12pt font having 8 pixel width and 12 pixel height) Special Effects (configuration file "tsWxGlobals" establishes defaults for various nCurses text markup styles that may be changed or overridden by the application programmer)

High-level API	Low-level API	Extended API
<p>Local Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		<p>Local & Remote Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell echo \$TERM # Display default type \$TERM=<xterm-family member ID> or <vt100-family member ID> echo \$TERM # Display changed type launch local application terminate local application login to remote platform host via "ssh <login ID>@<Host ID>:" launch remote application terminate remote application logout of remote platform host transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" terminate local shell logout of local platform host
	and Underline)	
<p>Color Palette (wxWidgets & wxPython)</p> <ul style="list-style-type: none"> 68-colors 4624-foreground/background color pairs 	<p>Color Palette (nCurses)</p> <ul style="list-style-type: none"> 8-colors (black, blue, cyan, green, magenta, red, white, yellow) 64-foreground/background color pairs 256-colors (option requires wide-character build of nCurses and Python Curses modules) 65536-foreground/background color pairs (option requires wide- 	<p>Color Palette (tsWxGTUI)</p> <ul style="list-style-type: none"> tsWxGraphicalTextUserInterface maps wxWidgets 68-colors (4624-color pairs) into nCurses 8-colors (64-color pairs) tsWxGraphicalTextUserInterface defines nCurses 256-colors (65536-color pairs) into wxWidgets 68-colors (4624-color pairs) and 188 other colors (60912 other color pairs) <p>Non-Color Palette (tsWxGTUI)</p>

High-level API	Low-level API	Extended API
<p>Local Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell launch local application terminate local application terminate local shell logout of local platform host 		<p>Local & Remote Command Line Interface Launcher</p> <ul style="list-style-type: none"> login to local platform host launch local shell echo \$TERM # Display default type \$TERM=<xterm-family member ID> or <vt100-family member ID> echo \$TERM # Display changed type launch local application terminate local application login to remote platform host via "ssh <login ID>@<Host ID>:" launch remote application terminate remote application logout of remote platform host transfer file(s) from remote platform host via "sftp <login ID>@<Host ID>:<File IDs>" terminate local shell logout of local platform host
	<p>character build of nCurses and Python Curses modules)</p> <p>Non-Color Palette (nCurses)</p> <ul style="list-style-type: none"> Platform-specific 1-color is visible for Foreground (White) and NOT visible (Black) for Background. The 1-color (2-color pair) is determined either by the phosphor (Green, Orange, White) used in the physical terminal's display or by the color adopted by the terminal emulator (Cygwin's console-based and xterm-based vt100 emulators use White on Black while Apple's Mac OS X xterm-based vt100 emulator uses Black on White) 	<ul style="list-style-type: none"> Platform-specific 1-color is visible for Foreground (White) and NOT visible (Black) for Background. The 1-color (2-color pair) is determined either by the phosphor (Green, Orange, White) used in the physical terminal's display or by the color adopted by the terminal emulator (Cygwin's console-based and xterm-based vt100 emulators use White on Black while Apple's Mac OS X xterm-based vt100 emulator uses Black on White)

Draft

15 APPENDIX C - DELIVERABLES

The following table describes the work product(s) to be delivered by the developer:

DELIVERABLE	DESCRIPTION
Software Package(s)	<p>The "tsWxGTUI_PyVx" Toolkit software:</p> <ul style="list-style-type: none"> ▪ Building Block Library Packages ("tsLibCLI" & "tsLibGUI") in Python 2x & Python 3x source code formats ▪ Tool Application Programs ("tsToolsCLI" & "tsToolsGUI") in Python 2x & Python 3x source code formats ▪ Test Application Programs ("tsTestsCLI" & "tsTestsGUI") in Python 2x & Python 3x source code formats ▪ Utility Modules ("tsUtilities") in Python 2x and Python 3x and POSIX-style shell script formats
Engineering Documentation ("tsEngineering") (Optional, Fee-based subscription)	<p>While the <i>TeamSTARS</i> "tsWxGTUI_PyVx" Toolkit is released as free, open source software, the engineering documentation establishes proprietary Copyright ownership by the original Toolkit Author(s).</p> <p>Toolkit recipients can obtain a license for the engineering documentation, if their intent is to assemble a large team to commercialize the software and want to kickstart the effort.</p> <p>The "tsEngineering" subdirectory contains a collection of Toolkit Author written large, complex documents, including structured documents, featuring multiple fonts and graphic images.</p> <p>The documents are authored using the following products:</p> <ul style="list-style-type: none"> ▪ Author-it® --- A help authoring tool (http://www.Author-it®.com) and content management system for creating, maintaining, and distributing single-sourced content. It generates Microsoft Word files with the appropriate outline numbering (for table of contents, sections, paragraphs, lists and figures) regardless of where the single-source content originated. ▪ Microsoft Office --- A collection of software applications (http://www.microsoftstore.com/store/msusa/en_US/cat/Office/categoryID.62684700) intended to be used by knowledge workers. The components are generally distributed together, have a consistent user interface and usually can interact with each other, sometimes in ways that the operating system would not normally allow. <p>Access (data base)</p> <p>Excel (spreadsheet)</p> <p>PowerPoint (presentation)</p> <p>Word (word processor)</p>

DELIVERABLE	DESCRIPTION
Software Package(s)	<p>The "tsWxGTUI_PyVx" Toolkit software:</p> <ul style="list-style-type: none"> ▪ Building Block Library Packages ("tsLibCLI" & "tsLibGUI") in Python 2x & Python 3x source code formats ▪ Tool Application Programs ("tsToolsCLI" & "tsToolsGUI") in Python 2x & Python 3x source code formats ▪ Test Application Programs ("tsTestsCLI" & "tsTestsGUI") in Python 2x & Python 3x source code formats ▪ Utility Modules ("tsUtilities") in Python 2x and Python 3x and POSIX-style shell script formats
	<ul style="list-style-type: none"> ▪ Microsoft Visio --- A software application (http://www.microsoftstore.com/store/msusa/en_US/list/Visio/categoryID.62687700) intended to be used by knowledge workers. Visio (diagrams) ▪ Fineprint Software (http://fineprint.com) FinePrint (A print driver, for Microsoft Windows, which allows you to manage the printing process. It helps you to reduce printing costs while enhancing and managing complex print jobs.) pdfFactory Pro (An application, for Microsoft Windows, that provides Adobe PDF compatible output.) <p>The documents accompany the computer software for the training and reference use of the software developer:</p> <ul style="list-style-type: none"> ▪ It explains the purpose, goals, non-goals, system requirements, interface requirements, software requirements, qualification requirements, development plan, software design, how it operates and how to install, use and troubleshoot it. ▪ It is provided in various application specific formats (such as Adobe PDF and Microsoft Office & Visio formats which may be read and edited by the free, open source, cross-platform LibreOffice Suite). <p>The "tsWxGTUI_PyVx" Toolkit software development and release documents are deliverable in Adobe's Portable Document Format and Microsoft's Word Format (with associated bit-mapped images in JPG or PNG Format):</p> <ul style="list-style-type: none"> ▪ Vol. 0 --- SDIST Announcement ▪ Vol. 1 --- SDIST Brochure ▪ Vol. 2 --- SDIST Introduction ▪ Vol. 3 --- SDIST Terms & Conditions ▪ Vol. 4 --- SDIST Software Development Plans ▪ Vol. 5 --- SDIST System Specification ▪ Vol. 6 --- SDIST Interface Requirements Specification ▪ Vol. 7 --- SDIST Software Requirements Specification ▪ Vol. 8 --- SDIST Release Notes ▪ Vol. 9 --- SDIST Software User's Manual

DELIVERABLE	DESCRIPTION
Software Package(s)	<p>The "tsWxGTUI_PyVx" Toolkit software:</p> <ul style="list-style-type: none"> ▪ Building Block Library Packages ("tsLibCLI" & "tsLibGUI") in Python 2x & Python 3x source code formats ▪ Tool Application Programs ("tsToolsCLI" & "tsToolsGUI") in Python 2x & Python 3x source code formats ▪ Test Application Programs ("tsTestsCLI" & "tsTestsGUI") in Python 2x & Python 3x source code formats ▪ Utility Modules ("tsUtilities") in Python 2x and Python 3x and POSIX-style shell script formats
	<ul style="list-style-type: none"> ▪ Vol. 10 --- SDIST Appendixes ▪ Vol. 11 --- SDIST Dictionary ▪ Vol. 12 --- SDIST To-Do List ▪ Vol. 13 --- SDIST Use Cases ▪ Vol. 14 --- SDIST Operator Documents
Engineering Notes	<p>The "tsWxGTUI_PyVx" Toolkit software development notes in Microsoft's Word Format:</p> <ul style="list-style-type: none"> ▪ Class List.doc ▪ Relationships_Between_tsWxGTUI_Toolkit_APIs.doc ▪ RGB to Color Name Mapping(Triplet and Hex).doc ▪ Writing a Python Package by Tarek Ziadé.doc <p>The "tsWxGTUI_PyVx" Toolkit software development notes in Microsoft's Access Format:</p> <ul style="list-style-type: none"> ▪ tsWxPython.mdb <p>The "tsWxGTUI_PyVx" Toolkit software development diagram notes in Microsoft's Visio Format:</p> <ul style="list-style-type: none"> ▪ Distributed System.vsd ▪ Networked Architecture.vsd ▪ System Architecture.vsd ▪ tsWxPython Org Chart.vsd <p>The "tsWxGTUI_PyVx" Toolkit software development notes in Microsoft's Excel Format:</p> <ul style="list-style-type: none"> ▪ Platform_Configuration.xls <p>The "tsWxGTUI_PyVx" Toolkit software development notes in Plain Text Format:</p> <ul style="list-style-type: none"> ▪ README_BMP.txt
Equipment Operator Documentation ("tsDocuments")	<p>The "tsWxGTUI_PyVx" Toolkit installation, configuration, operation and troubleshooting documents in Plain Text Format:</p> <ul style="list-style-type: none"> ▪ tsDocCLI-1-GettingStartedFiles ▪ tsDocCLI-2-DistributionReadMeFiles ▪ tsDocCLI-3-DeveloperReadMeFiles ▪ tsDocCLI-5-UsageTermsAndConditions ▪ tsManPages ("tsLibCLI", "tsLibGUI", "tsToolsCLI", "tsToolsGUI", "tsTestsCLI",

DELIVERABLE	DESCRIPTION
Software Package(s)	<p>The "tsWxGTUI_PyVx" Toolkit software:</p> <ul style="list-style-type: none"> ▪ Building Block Library Packages ("tsLibCLI" & "tsLibGUI") in Python 2x & Python 3x source code formats ▪ Tool Application Programs ("tsToolsCLI" & "tsToolsGUI") in Python 2x & Python 3x source code formats ▪ Test Application Programs ("tsTestsCLI" & "tsTestsGUI") in Python 2x & Python 3x source code formats ▪ Utility Modules ("tsUtilities") in Python 2x and Python 3x and POSIX-style shell script formats
	<p>"tsTestsGUI")</p> <p>1. Operator Documentation</p> <p>User documentation for each "tsWxGTUI" Toolkit distribution is contained, in plain text format, in the subdirectory named ".tsDocCLI". The subdirectory is organized, by topic, to contain the following:</p> <p>a. How to Prepare Platform to Get Started</p> <p>The ".GettingStartedFiles" subdirectory contains the following file(s):</p> <p>"README-GettingStarted.txt" ---</p> <p>b. How to Install and Redistribute</p> <p>The ".DistributionReadMeFiles" subdirectory contains the following file(s):</p> <ul style="list-style-type: none"> ▪ "AUTHORS.txt" --- List of the principal "tsWxGTUI" Toolkit author(s) and authors credited for work covered by a prior copyright and license. ▪ "BUGS.txt" --- List of Known Problems / Issues. ▪ "CHANGE_LOG.txt" --- List of Additions, Modification and Deletions. ▪ "CONFIGURE.txt" --- Instructions for applying factory and site-specific configurations. ▪ "COPYING.txt" --- Instructions for copying all or a portion of the distribution. ▪ "FAQ.txt" --- Answers to Frequently Asked Questions. ▪ "INSTALL.txt" --- Describes steps to download, extract install and configure the "tsWxGUI" Toolkit. ▪ "LICENSE.txt" --- General and special arrangements, provisions, rules, specifications and standards that form an integral part the agreement or contract between the creator and recipient of Copyrighted and Licensed Work. ▪ "MANIFEST.txt" --- Tally List for deliverable items. ▪ "NEWS.txt" --- Announcements of new releases. ▪ "NOTICES.txt" --- Details the copyright(s) and license(s). ▪ "OPERATE.txt" --- Describes steps to use the "tsWxGUI" Toolkit. ▪ "README.txt" --- Introduces new recipients to the purpose, goals, non-goals, design and features of the computer software product. ▪ "THANKS.txt" --- Acknowledgments to those otherwise unsung heroes who contributed time and effort to supporting the authors as planners, editors, designers, coders and testers.

DELIVERABLE	DESCRIPTION
Software Package(s)	<p>The "tsWxGTUI_PyVx" Toolkit software:</p> <ul style="list-style-type: none"> ▪ Building Block Library Packages ("tsLibCLI" & "tsLibGUI") in Python 2x & Python 3x source code formats ▪ Tool Application Programs ("tsToolsCLI" & "tsToolsGUI") in Python 2x & Python 3x source code formats ▪ Test Application Programs ("tsTestsCLI" & "tsTestsGUI") in Python 2x & Python 3x source code formats ▪ Utility Modules ("tsUtilities") in Python 2x and Python 3x and POSIX-style shell script formats
	<ul style="list-style-type: none"> ▪ "TO-DO.txt" --- A To-Do-List provides a roadmap for development and troubleshooting work. ▪ "TROUBLE SHOOT.txt" --- Provides a list of available reference resources and a guide for planning, developing and troubleshooting a cross-platform system of hundreds of files each containing a few, tens or hundred of class, data and method definitions. Its complexity becomes apparent in the recent software Lines-Of-Code metrics. <p>c. How to Use and Modify</p> <p>The "./DeveloperReadMeFiles" subdirectory contains the following file(s):</p> <ul style="list-style-type: none"> ▪ "README.txt" ▪ "README_1st.txt" ▪ "README_1st-PyPI-Dev-tsWxGTUI.txt" ▪ "README-01-Title_Page.txt" ▪ "README-02-Table_Of_Contents.txt" ▪ "README-03-Purpose.txt" ▪ "README-04-Goals.txt" ▪ "README-05-Non-Goals.txt" ▪ "README-06-Design_Strategy.txt" ▪ "README-07-Design_Architecture.txt" ▪ "README-08-Software_Configuration_Management.txt" ▪ "README-09-tsWxGTUI_Directories.txt" ▪ "README-10-tsToolkitCLI_Directories.txt" ▪ "README-11-tsToolkitGUI_Directories.txt" ▪ "README-12-Features.txt" ▪ "README-13-Current_Capabilities.txt" ▪ "README-14-Current_Limitations.txt" ▪ "README-15-Reference_Documents.txt" <p>d. How to Learn "tsWxGTUI" Toolkit Software Development</p> <p>The "./DeveloperTutorialFiles" subdirectory contains the following file(s):</p> <ul style="list-style-type: none"> ▪ "CLI_0_hello_world_print_statement.py"

DELIVERABLE	DESCRIPTION
Software Package(s)	<p>The "tsWxGTUI_PyVx" Toolkit software:</p> <ul style="list-style-type: none"> ▪ Building Block Library Packages ("tsLibCLI" & "tsLibGUI") in Python 2x & Python 3x source code formats ▪ Tool Application Programs ("tsToolsCLI" & "tsToolsGUI") in Python 2x & Python 3x source code formats ▪ Test Application Programs ("tsTestsCLI" & "tsTestsGUI") in Python 2x & Python 3x source code formats ▪ Utility Modules ("tsUtilities") in Python 2x and Python 3x and POSIX-style shell script formats
	<ul style="list-style-type: none"> ▪ "CLI_1_hello_world_print_function.py" ▪ "CLI_2_hello_world_script_environment.py" ▪ "CLI_3_hello_world_main_module_application.py" ▪ "GUI_4_Curses_LowLevel_WidgetApi_application.py" ▪ "GUI_5_tsWxGTUI_HighLevel_WidgetApi_application.py" ▪ "GUI_6_tsWxGTUI_HighLevel_BoxSizerApi_application.py" ▪ "ReadMe_Developer_Tutorial_Files.txt" ▪ "test_tsCxGlobals.py" ▪ "test_tsWxGlobals.py" ▪ "tsCxGlobals.py" <p>e. Terms & Conditions</p> <p>The "./UsageTermsAndConditions" subdirectory contains the following file(s):</p> <ul style="list-style-type: none"> ▪ "COPYRIGHT.txt" ▪ "LICENSE.txt" ▪ "NOTICES.txt" ▪ "SplashScreenDesignersGuide.txt"