

Teradyne CTS TPS Database

A Major Qualifying Project Report:

submitted to the faculty of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by:

Jonathan Marokhovsky

Date: August 29, 2012

Approved:

Professor Gary F. Pollice, Major Advisor

1. Workflow
2. Data driven
3. State machine

Abstract

This project implements a system for Teradyne Corp. that provides a workflow for their Quality Assurance Engineers' Test Program Set (TPS) verification process. To make the system flexible and accessible regardless of what software is available to the engineers, the Spectrum Common Test System (CTS) Database is designed to work using web browsers and uses a centralized database at its heart. Contained in this database is a data-driven state machine which controls the path of the TPS logs. This creates the system which can replace Teradyne's tedious spreadsheet tracking system with a modern process flow and detailed tracking. The CTS Database became a reality through the use of JavaServer Pages (JSP), Java, Expression Language (EL), Oracle Structured Query Language (SQL), and JavaScript.

Acknowledgements

Acknowledgements

Teradyne:

- Tarra Marchetti – Organized this project and set up onsite meetings
- Luis Villalta – Firsthand Engineer’s input and help with setting up the application.

WPI

- Gary Pollice, project advisor.
- Nicholas Deapen

Table of Contents

ABSTRACT	I
ACKNOWLEDGEMENTS.....	II
LIST OF ILLUSTRATIONS.....	III
LIST OF TABLES	IV
1. INTRODUCTION	1
2. BACKGROUND	3
3. METHODOLOGY	4
3.1. ARCHITECTURE.....	4
3.2. PROCESS FLOW.....	5
3.3. STATE MACHINE.....	6
3.4. DATABASE DESIGN.....	9
3.4.1. <i>Transactional Database</i>	9
3.4.2. <i>Third Normal Form</i>	9
3.4.3. <i>ID Number Sequences</i>	10
3.4.4. <i>Using Triggers to Log all Tables which Need History</i>	10
3.4.5. <i>Priming Insert Statements</i>	10
3.5. LOGIN AND SECURITY	11
3.5.1. <i>The Password Changer</i>	11
3.6. USER PRIVILEGES	12
3.7. WEB DESIGN	14
3.7.1. <i>Dynamic Pages</i>	14
3.7.2. <i>Expression Language (EL)</i>	14
3.7.3. <i>Using JavaScript for form validation</i>	14
3.8. THE JAVA LAYER	15
3.8.1. <i>The Database Connection</i>	15
3.8.2. <i>Managers</i>	16
3.8.3. <i>Factories</i>	16
3.8.4. <i>Why Java?</i>	17
3.8.5. <i>Java Beans and Bean Factories</i>	17
4. RESULTS AND ANALYSIS	17
5. FUTURE WORK AND CONCLUSIONS	18
5.1. FUTURE WORK	18
5.2. CONCLUSIONS	19
6. APPENDIX A: INSTALLATION DOCUMENTATION.....	20
7. APPENDIX B: MAINTENANCE DOCUMENTATION.....	31
8. APPENDIX C: USER DOCUMENTATION	46
9. GLOSSARY.....	74
10. REFERENCES.....	75

List of Illustrations

Figure 1: TPS Overall Architecture.	4
Figure 2: System Interaction Diagram.	5
Figure 3: State Machine Entity Relationship Diagram.	7
Figure 4: The Flow of the Login Screen.....	11

List of Tables

List of Tables

Table 1: The first three states in the States table.	8
Table 2: A Description of User Privileges.	13

1. Introduction

This project involves creating a database system that automates the process of verifying the creation or changes of Test Program Sets (TPSs). Teradyne uses TPSs to adapt their Spectrum Common Test System (CTS) to different airplane parts. Teradyne must constantly verify that their TPSs align with the parts they are designed to test. It is the logs of this verification process which the results of this project will replace. Currently at Teradyne this involves three Excel spreadsheets which require engineers to manually input timestamps and copy and paste information throughout. The CTS Database combines and centralizes all the information in those three spreadsheets and automatically timestamps the process to remove as much human error as possible. The flow of TPSs is programmed into the database to make it easy to modify should the process change since TPSs all follow a static process flow. The design is entirely data-driven with rules and conditions which makes it malleable. A CTS Database administrator can easily change the process flow with existing database tools by inserting and updating the metadata stored in the database.

Most projects do not run as smoothly as they were originally planned to and this project was almost as far from an exception to that rule as possible. When this project started I knew very little about Web programming and using servlets to make the web programs dynamic. My area of expertise before this project was in Java with some experience in programming databases. After my partner left the project I quickly had to fill the gaps in my knowledge and had to teach myself to know this project completely, even though the entire project had been overhauled just prior to my partner's departure to address some design shortcomings. Even with all of these setbacks I managed to make a system which suited Teradyne's needs.

Introduction

The sections to come in this report are:

- Background – A more in-depth description of Teradyne and the problem this project addresses.
- Methodology – How the main problem is approached, mainly focusing on the state machine and how it affects the rest of the system.
- Results and Analysis – How well the system meets the initial requirements and how it meets expectations.
- Future Work and Conclusions – A description of any work which could further improve the experience the CTS Database provides and any closing thoughts.

2. Background

Teradyne is a leading supplier of automated test equipment. They develop and sell equipment which tests complex electronics ranging from hard disk drives to military and aerospace parts. The organization within Teradyne sponsoring this project is the Quality Assurance (QA) group. This organization tests the equipment meant to test other complex equipment. The problem the Teradyne QA engineers have is that their test data logging process is not as reliable as it could be. Teradyne engineers need a system that can automatically timestamp items, let them add notes on different parts, and send Test Program Sets (TPSs) down a specified path. Other systems that engineers looked into are not specialized enough to support what the engineers want to do and not all of the systems that did fit their criteria allow for group collaboration.

3. Methodology

3.1. Architecture

The TPS application’s design is based on the Model-View-Controller (MVC) paradigm: a design pattern that separates the display of content from the logic used to obtain and manipulate that content (Christensen 2010). In the case of this system, the View is the User Interface (UI), the part of the system which the user sees. The Model is the database which stores all the information. The Controllers are all of the Servlets that run to support a user’s actions on the database. This architecture employs a powerful metadata driven state machine for ease of use and flexibility to change the flow and rules for the process flow of the entire system.

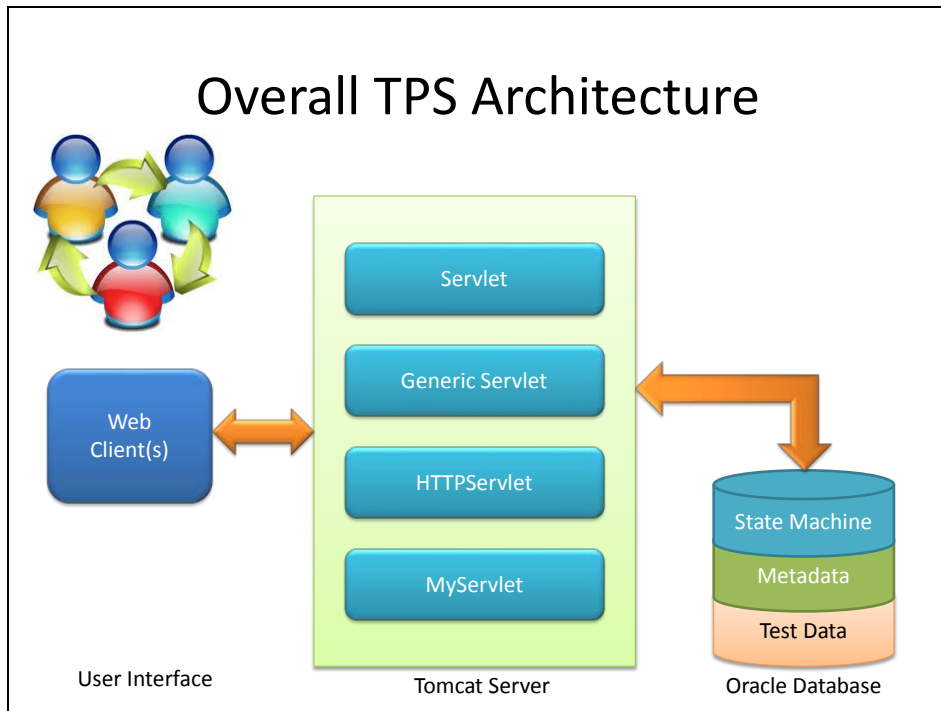


Figure 1: TPS Overall Architecture.

3.2. Process Flow

The CTS TPS application is a workflow application where users sign in to create TPSs, assign tasks to the appropriate individuals, and track the progress until completion. The system is built with the Basic User in mind (see the Basic Users section) whose general workflow is repetitive and follows a set of steps and conditions to bring a TPS from its creation to its completion. Figure 2: System Interaction Diagram illustrates the general flow driven by a Basic User's choices and the state machine's possible outcomes.

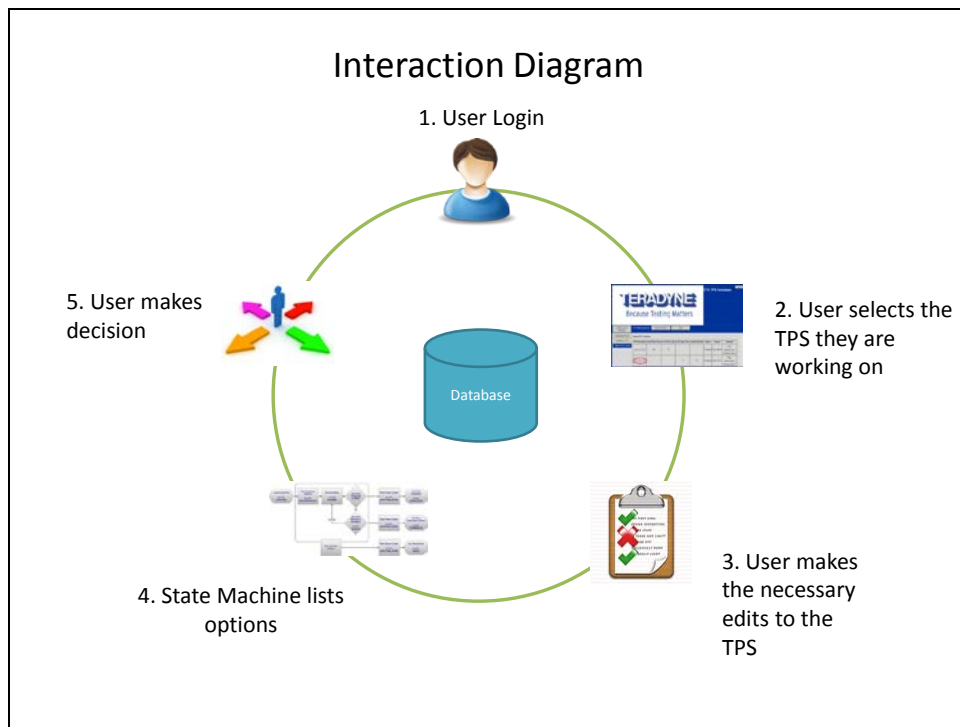


Figure 2: System Interaction Diagram.

The system is built on a relational database which contains the state machine. The state machine takes a user's interaction on a TPS and uses that information to determine the possible next steps for the user's TPS. Users interact with the Java web client which interfaces with multiple servlets that serve as clients to the state machine. The logic, rules and conditions of the

Methodology

state machine are not maintained in Java, but instead consist of metadata in the relational database. The role of the servlets is to translate the user information into input parameters to the state machine then translate the output of the state machine into a form that can be displayed to the user so the user can make a selection or acknowledge state change and start the cycle anew. The following is a step-by-step breakdown of this process flow.

Step 1: User login. User has received a notification assigning him/her for a new task or the user has test results to input in the system to update the system and pass it on to the next state.

Step 2: User selects the TPS to work on. Or the User creates a new TPS (not shown in diagram).

Step 3: User makes the necessary edits based on their results.

Step 4: State machine lists options for the next state for the TPS.

Step 5: User makes decision. The TPS now goes to the next user who owns it and the process begins again from Step 1.

3.3. State Machine

The state machine is the core of the system; driving the process flow, paths, and conditions for moving from one state to another. Metadata stored in the relational database drives the system and gives system developers the ability to modify the process flow without needing to

change any of the program code (see Figure 3: State Machine Entity Relationship Diagram.).

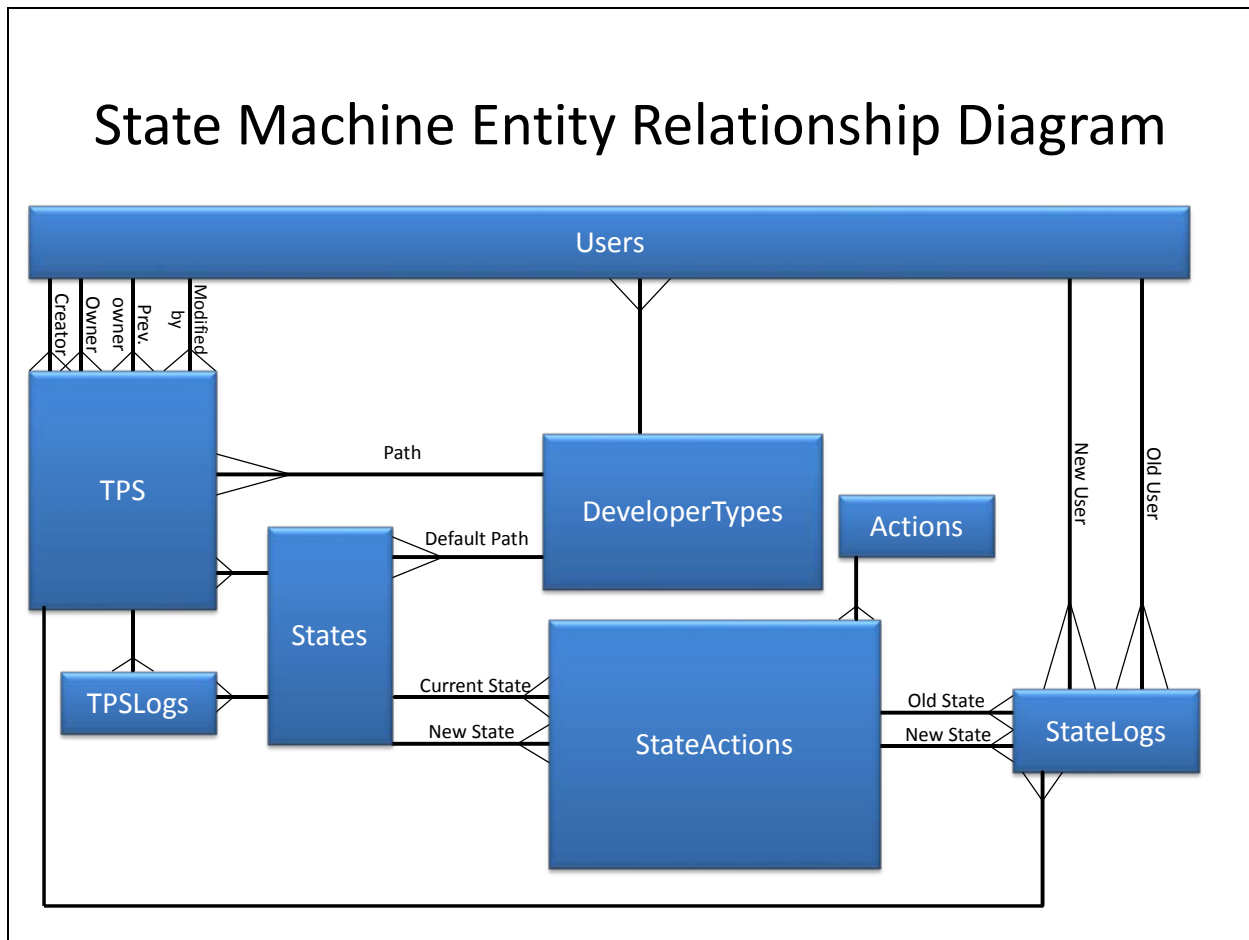


Figure 3: State Machine Entity Relationship Diagram.

The state machine consists of two driving tables: States and StateActions. The ‘States’ table defines each of the possible states of the process flow. States are numbered sequentially starting from 0. State 0 is the starting state for all newly created TPSs; it is never visible to users because the system automatically forwards the TPS to its next state depending on the TPS’s path given by the user when the TPS was created. Every step after the first is directly driven by a user’s selected action.

The ‘StateActions’ table controls which states a TPS can progress to. Each potential state change (e.g. moving from State 0 to State 1) is defined by a single row in the StateActions table.

Methodology

It is possible to have more than one state to progress to from a single state; for example from State 0 it is possible to move to either State 1 or State 2 depending on the developer type that is assigned the TPS record which stands for the path. Other state changes are driven by a user's chosen action which comes from the Actions table. The StateActions table provides the mechanism to specify the driving table, driving column, and driving value in order to offer flexibility for the various paths a TPS record can take. For example to proceed from State 1 to the next state, the system knows the user must select from a list of specific actions. A Servlet queries for all rows in StateActions where 'current_state' is 1 which returns two rows driven by the Actions table. The servlet then creates a query on the fly to display the possible actions to a user. This drives the next possible state according to the resulting possible values retrieved on the SQL statement is executed.

Table 1: The first three states in the States table.

State ID	State Name	Path	Description
0	Created but Without Developer	N/A	Initial state of all TPSs where the Developer is chosen.
1	Created	Ter Apps	This is the first state after TPS is created with Developer specified. The Developer is Ter Apps in this case.
2	Created	ASP	This is the first state after TPS is created with Developer specified. The Developer is ASP in this case.

The design of this state machine allows users to create new driving tables and column-value-pairs to shape the flow of the process according to the needs of the end users. SQL

Methodology

statements are generated during the execution of each step for maximum flexibility and choices of outcomes.

The next tables are the Actions and DeveloperTypes tables which govern the current possible conditions for each TPS state change. The TPS table is where the various attributes of a TPS record are persisted and also where the current state and the current owner are recorded.

3.4. Database Design

3.4.1. Transactional Database

For peak stability, the database connection is completely transactional. This means for every call to the database a new connection is created to the database, the command is processed in the database, the result is given, and then the connection is closed. The advantage of this is it is very hard to put the database in an unstable state due to a connection being interrupted since the actual connection times are so short. The disadvantage of this is for every connection to the database the user must wait for the connection to occur which is affected by the client's physical connection to the database. This makes the system respond slower than a constant connection however stability outweighs speed in the case of this system.

3.4.2. Third Normal Form

The backing database is designed to be maintained by Teradyne's IT department. As such, the IT department requires an Oracle 10g database in Third Normal Form (3NF). This means each attribute in a table had to represent information unique to its given row (Chapple 2012). For example the Notes table has a note Identification number (ID), its content, a timestamp, a foreign key for the TPS the note relates to, and a foreign key referencing the user

Methodology

that created the note. If the creator's foreign key was instead the creator's name and that user's email the Notes table would violate 3NF because the table is no longer completely about Notes; it is instead partially about the creator. 3NF forces the database to be modular, a common theme in the design of the CTS Database. This means it is possible to change a table without the change affecting other tables in the database.

3.4.3. ID Number Sequences

Every object has its own unique ID in order to make passing and finding objects in the system easier. This allows a user to have the same name as another user without causing the database to throw an error. There is an ID sequence for each main object (Users, TPSs, and Notes) generated in the database. This prevents synchronization errors between clients since the unique ID is handled in a centralized location.

3.4.4. Using Triggers to Log all Tables which Need History

Every table which needs to be logged has its own trigger for every necessary action. An example of this is the State Update trigger, which records the information from the old TPS state and the new information coming in into another table named the StateLogs table when the new TPS information contains a state different from the old information.

3.4.5. Priming Insert Statements

When the system's database is first created the system will not work without first having its database primed because many parts of the system are data-driven. An example of this is the fact that this is a user-run system, but upon initial installation of the schema the Users table is empty. For this reason there are primary insert statements in the database schema script that populate all of the parts of the database upon installation.

3.5. Login and Security

A good example of how the whole system works together is the login screen, Figure 4.

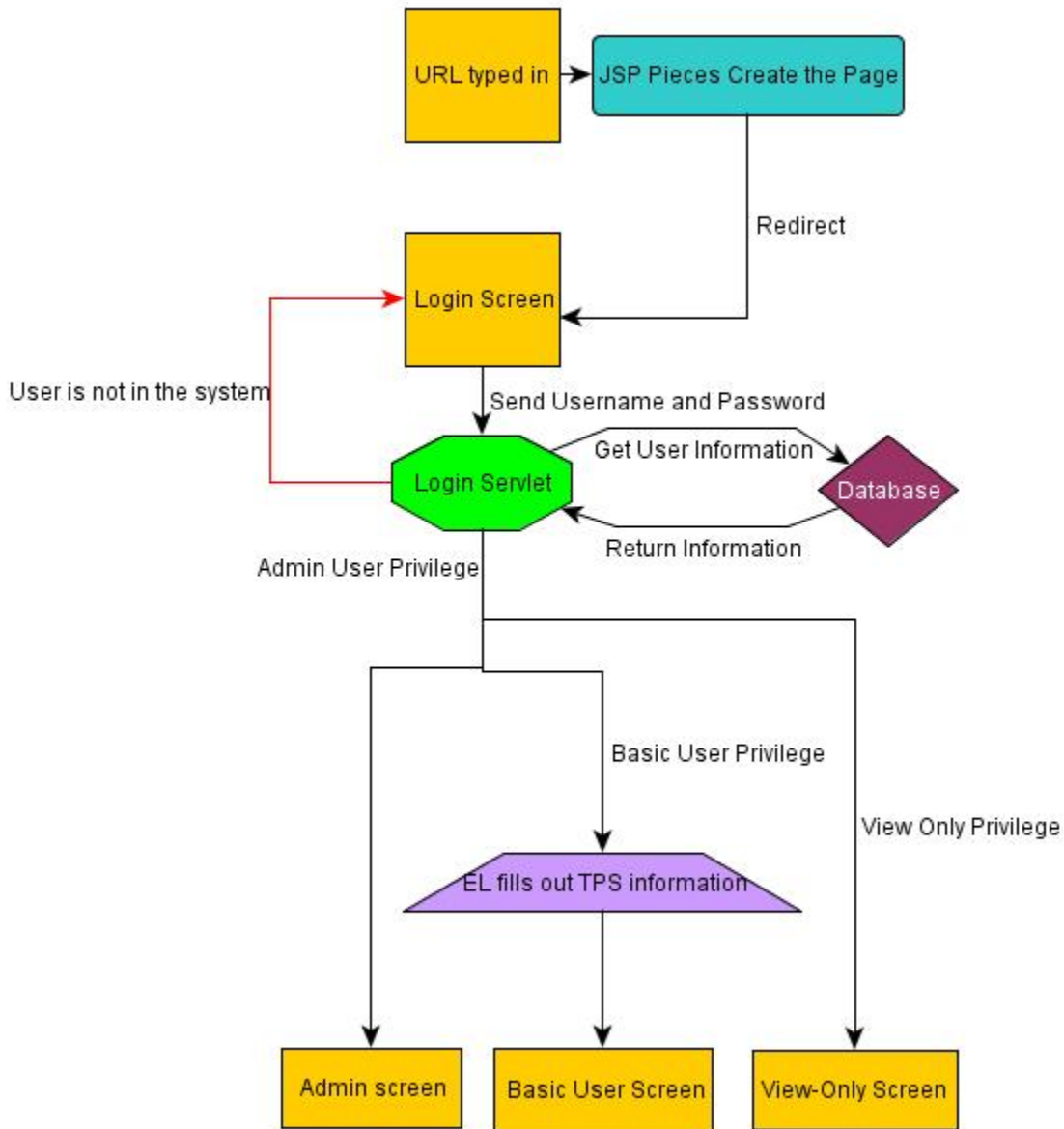


Figure 4: The Flow of the Login Screen

3.5.1. The Password Changer

A temporary password changer is in place to enable Administrator users to change the passwords of users in the system. Other users in the system must seek out an Administrator in

Methodology

order to change their passwords because of this. This functionality is easily removable for Teradyne because this responsibility should be handled by Teradyne's Login Service mechanism.

3.6. User Privileges

The following table, Table 2, provides a basic overview of the User Privilege Levels available to users of the CTS TPS Database system. For a more in-depth description of their available functions see Appendix C: User Documentation: the User Documentation which is supplied with the system.

Privilege Level	Abilities in the CTS Database	Best Suited For
Administrator	<ul style="list-style-type: none"> • Adding and Removing users. • Managing users' privileges. • Managing users' passwords. • Managing TPSs regardless of those TPSs' owners. • Correcting issues in TPSs. • Change TPS Owners without changing the TPS's state. • Change TPS States without abiding by the state machine. • All the abilities of the other two privilege levels. 	<ul style="list-style-type: none"> • Senior engineers who have had extensive experience with TPSs
Basic User	<ul style="list-style-type: none"> • Create new TPSs. • Progress TPSs which the user owns through the state machine. • Add notes to TPSs which the user owns. • Change values of TPSs which the user owns. • All actions which a View-Only User can do. 	<ul style="list-style-type: none"> • Engineers who deal with TPSs.
View-Only User	<ul style="list-style-type: none"> • View information on TPSs in the system. • Search the system for certain TPSs. 	<ul style="list-style-type: none"> • Managers looking for inefficiencies in the system or want to make sure engineers are working as expected.

Table 2: A Description of User Privileges.

3.7. Web Design

3.7.1. Dynamic Pages

JavaServer Pages (JSP) allows the User Interface (UI) to be made out of common parts (“Java EE Tutorial”). This, coupled with the use of Expression Language (EL), make sure the UI follows the theme of modularity present throughout the CTS Database.

3.7.2. Expression Language (EL)

A drawback to using JSP is the inconsistent behavior of embedding Java in JSPs via JSP scriptlets. The JSP 2.0 protocol has a solution for this: Expression Language (EL). EL is a tag-based language which removes the need for scriptlets in JSP by hard-coding values at compile time in JSP pages. Converting pages to use EL instead of scriptlets allows the system to keep the functionality of the scriptlets while removing the risk of having live code on the page (Lubke, Ball, and Delisle). Despite not having scriptlets, the system can still dynamically choose how many tabs to display in the header according to a user’s privilege level among other things. This makes EL extremely valuable throughout the pages of the system.

3.7.3. Using JavaScript for form validation

JavaScript in the system validates any user input at the form level. It makes sure that no information is sent to the server that could break the system and makes sure that mistakes can be fixed by the user before a form’s input is committed. Form-validation is compartmentalized by functions that are used as validation rules to make it easy to remove a rule if necessary once again adhering to the theme of modularity. Each rule has a single type of check. For example, on some forms there are required fields that cannot be left blank. There are other fields on the same

Methodology

form that can only be numbers. As a result there is a single rule to make sure all the fields labeled to only have numbers are checked as such and there is another rule to make sure all required fields are checked and made sure they are filled out. The check scripts use the “on submit” action to make sure the checks are run before the form submits and will only allow the form to submit if all the checks pass. This makes it very easy to add more rules or remove unnecessary rules in the future.

One example of this is a rule to check that either two fields of one type were filled out or two fields of another type were filled out. All four fields could not be filled out nor could three fields be filled out nor could a single field. When engineers at Teradyne reviewed the system, it was found that those rules are not necessary. Engineers could review how the system worked without the rule in the amount of time it took to reload the page because of the compartmentalized rules.

3.8. The Java Layer

The original documentation from Teradyne coupled with the available skillset of the developers made it easy to begin designing the system in the early planning stages of the project. Java is a forte of both of the original designers and a web design is flexible and portable. The only remaining variable is the database. With the object-oriented Java language, the database could be left as a variable, designed as an interface, and implemented later.

3.8.1. The Database Connection

The system code linking the Java code to the database is abstracted using Interfaces and factories to keep with the theme of a modular system. The current code supports Oracle 10g but

Methodology

the system is designed in such a way to allow for a simple transition to a different database if need be by implementing another version of the database interfaces available in the system.

3.8.2. Managers

Every part of the CTS Database which must interact with another non-Java portion of the System must go through a Manager interface. Each Manager manages and interprets information from a connection to a non-Java part of a system. For example the database connection is managed by the Oracle Database Manager. In the future if Oracle is no longer supported at Teradyne, developers can make a MySQL Database Manager and as long as it instantiates the original Database Manager interface, no other parts of the system would have to be tampered with. Part of what makes the Managers work with the CTS Database is the Factories which act as interfaces between the Database Manager level and the rest of the system.

3.8.3. Factories

Factories in this system are full objects that handle the creation of certain parts of the system. A good example is the TPS Factory class that handles every action to do with TPSs. When the TPS Factory class is first made it takes in a Database Manager which is what it uses for all database actions such as getting a whole TPS from the database or sending an entire TPS to the database to be stored. The TPS Factory does all operation on a TPS that can be done in Java without consulting the database. This keeps the amount of commands to instantiate when changing databases at a minimum and minimizes the chance that changing a database will be incompatible.

Results and Analysis

3.8.4. Why Java?

Both developers knew how to program in Java. Also the object-oriented paradigm makes organizing the system simple for anyone who inherits the code base. Finally, Java makes it easier to send information to the two ends of the system – the database end (using JDBC) and the Web end (using Java Beans).

3.8.5. Java Beans and Bean Factories

Java Beans are Java objects which are optimized in passing information. They only use getter and setter methods and must conform to a particular convention (“JavaBeans”). This makes them very powerful when it comes to sending organized information to a front end (“Java EE Tutorial”). Another plus is EL can easily interpret Java Beans making complex information transfer simple. The only problem with Java Beans is the fact that information cannot be sent to the bean through a constructor. Bean Factories solve this dilemma. Bean Factories convert the information from a complex object (one with a constructor to automatically load it) into Beans. For example there is a complex User object which is passed around within the system and there is a much less complex User Bean which is sent to the front end when necessary. There is a User Bean Factory that converts from the User object to the User Bean object. This means a programmer who needs to send a user’s information to the front end must only get the User object, convert it to a Bean, and send that Bean to the UI.

4. Results and Analysis

When this project began it seemed simple: make a database and then make a website to put on top of it with some logic between the two to make the process flow work as directed. All of the little parts of the system which had to be made and accounted for made the project much

Future Work and Conclusions

more complicated. When some requirements were not followed and the system was built with an object database instead of an Oracle database the project became much more difficult and after a complete redesign and a lost partner difficulty peaked. After learning how to utilize JSP, Java servlets, Web servers, Oracle databases, and EL the system took shape. The system became organized so that developers in the future do not have the same difficulties. The resulting data-driven state machine allows Teradyne to progress their TPSs from creation to closing automatically and reliably.

This project did not go as expected. But that does not make it any less of a success. Working with Teradyne showed me that requirements are destined to change the importance of designing software in a modular fashion to help cope with changing requirements and prevent the loss of precious time switching out pieces of the system. In the end, Teradyne is receiving a system which will solve their TPS logging problems smoothly.

5. Future Work and Conclusions

5.1. Future Work

There are several parts of the system which became requirements but had to be put aside due to time constraints or lack of information. The most pressing and the most likely to be worked on soon is the fact that the system is not connected to Teradyne's login service. This was put aside because information on how to connect to that service was not available but it is a higher priority requirement because it puts the system at a heightened security risk. With this update comes another topic of future work – linking the system to Teradyne's email service. This was in the original specifications for the system but, like the login service, information on connecting to the service could not be attained. This capability of the system would be very

Future Work and Conclusions

useful to have though as it could make it easier for TPSs to be passed from engineer to engineer and there are also some standardized emails which would be automatically sent for handoffs.

Some other future work on this includes a protocol which allows users to attach files to TPSs and more work on improving the GUI.

5.2. Conclusions

The goal of this project is to deliver to Teradyne a workflow system to improve the productivity of their test engineers. Teradyne has a well-defined process to test various parts but they are challenged to follow the process methodically. As a result, they asked WPI to look into automating the steps of the process to reduce errors and missteps that affect their effectiveness. The project started with taking their requirements and interpreting them into a computer system automating and guiding the engineers during each step of the test cycle. The project took a life of its own with significant challenges along the way. After design changes and bug fixes Teradyne is receiving a system that will automatically progress TPSs down a flow and dynamically choose which states the TPS will be able to go to next. In the ever changing market a company needs methods of making sure their product is validated reliably that is adaptable enough to evolve with the company, this software addresses both issues.

6. Appendix A: Installation Documentation

WORCESTER POLYTECHNIC INSTITUTE

Installation Documentation

Teradyne CTS TPS Database

Version 1.0.2

Jonathan Marokhovsky

8/11/2012

Contents

DELIVERABLE LIST	22
DATABASE SCHEMA INSTALLATION	23
DOWNLOADING AND INSTALLING ECLIPSE	24
IMPORTING THE TERADYNECTSDATABASE.ZIP FILE INTO ECLIPSE	25
CTS SYSTEM DATABASE LINKING	27
DEPLOYING TO THE TOMCAT SERVER	29
CREATING THE FIRST USER	30

Deliverable List

- The *TeradyneCTSDeliverable.zip* file which contains:
 - *Installation Documentation.docx*
 - *User Documentation.docx*
 - User help with in depth directions on how to do all of the functionality which a User of the Teradyne CTS Database system might need to use.
 - *Maintenance Documentation.docx*
 - Directions on where to look to add new functionality to the Teradyne CTS Database system and tips on how to maintain the code base.
 - *database.sql*
 - This is the SQL script file which will build the database schema which makes up the backbone of the Teradyne CTS Database system.
 - *TeradyneCTSDatabase.zip*
 - This is the code of the rest of the system exported into a *.ZIP* file. Directions on unpacking this *.ZIP* file into Eclipse are in [IMPORTING THE TERADYNECTSDATABASE.ZIP FILE INTO ECLIPSE.](#)

Database Schema Installation

These are directions for building the Oracle Database Schema. This is the backbone for the whole CTS System. These instructions will customize an empty database so that it can work with the CTS System.

Prerequisites

- Have an empty Oracle database created for the purpose of this system.
- Have the *database.sql* file for the CTS Database System. This should be in the *TeradyneCTSDatabase.zip* file received which contains the CTS System.
- A User for the database with **CREATE** privileges.

Steps

- 1) Run the *database.sql* file as a script on the database. If possible pipe the output of the script to a text file to make the next step easier.
- 2) Check over the output of the script and make sure there were no errors. There will be four warnings which say “SP2-0272: escape character cannot be alphanumeric or whitespace”, these can be ignored.

Downloading and Installing Eclipse

These are the steps to install the Eclipse IDE so that it will work in importing from the *TeradyneCTSDatabase.zip* file correctly.

NOTE: The Eclipse IDE is constantly being updated. The system was developed using Eclipse 3.7 (Indigo) and at the time of writing this the Eclipse version at Eclipse.org is Eclipse 4.2 (Juno). The idea behind the directions will still work but where to find the commands may change.

Prerequisites

- Know whether the system you are working on has a 32 bit operating system or a 64 bit operating system.

Steps

1. Go to <http://www.eclipse.org/downloads/>.
2. Look for the *Eclipse IDE for Java EE Developers* and download the appropriate bit version. You should be downloading a *.ZIP* file.
3. Extract the given *.ZIP* file to a place you will remember. This should extract an *eclipse* directory to this place.
4. Open the *eclipse* directory where you can find the *eclipse.exe* file. Clicking this will start up Eclipse.
5. The first time Eclipse is started it will ask for a workspace, either choose a location on your own pressing the Browse button or stick with the default. After loading this should

Importing the TeradyneCTSDatabase.zip File into Eclipse

come up with a screen. Press the Workspace button and it should show the main coding space. This step should not have to be repeated after the first time starting up.

Importing the TeradyneCTSDatabase.zip File into Eclipse

These directions are to import the *TeradyneCTSDatabase.zip* file into Eclipse to create the appropriate class structure automatically.

Prerequisites

- Have the *TeradyneCTSDatabase.zip* file.
- Complete the [DOWNLOADING AND INSTALLING ECLIPSE](#) section to install Eclipse.

Steps

1. Start Eclipse.
2. Make a new Project
 - a. Right click the Project Explorer section usually on the Left side of the screen.
 - b. Select New > Project...
 - c. Scroll down to Web > Dynamic Web Project double click on it.
 - d. For a "Project name" make it TeradyneCTSDatabase.
 - e. Press Finish.
3. Right click the new Project named *TeradyneCTSDatabase*.
4. Go to File > Import.
5. Select General > Archive File.
6. Browse for the *TeradyneCTSDatabase.zip*.
7. Make sure the "Into folder" field reads TeradyneCTSDatabase.

Importing the TeradyneCTSDatabase.zip File into Eclipse

8. Press Finish and it should all be imported correctly.

CTS System Database Linking

Prerequisites

- Completed the Database Installation section.
- Have the database name, hostname, and a port number for the database made in the [DATABASE INSTALLATION](#) section.
- Have an Application User solely for the system to connect to the database which has **INSERT** and **UPDATE** privileges.
- Have the source files for the CTS System. This should be in the form of a [.ZIP](#) file.
- Have the Eclipse Enterprise Edition IDE installed on a computer, or some other way to modify Java files.
 - Another IDE which can convert a Java Project to a [.WAR](#) file could work but some steps might be different. The following steps describe how to perform this action using Eclipse.

Steps

- 1) Start Eclipse and make sure it has access to the TeradyneCTSDatabase project
- 2) Open the TeradyneCTSDatabase project in the Project Explorer.
- 3) Under *Java Resources* navigate to the *src* folder.
- 4) Navigate to [com.teradyne.tpsdb.database.OracleDatabaseConnector.java](#).
- 5) Find the class variables **DEFAULT_HOST**, **DEFAULT_DB**, and **DEFAULT_PORT** and modify them to be the hostname, database name, and port number respectively to reflect the database which the [database.sql](#) script was run on.

CTS System Database Linking

- 6) Find the class variables **UN** and **PW** and modify them to be the Application User name and password respectively.
- 7) Save your changes.
- 8) Right click on the *TeradyneCTSDatabase* project and go to Export > WAR file.
- 9) Make sure the Web project field reads TeradyneCTSDatabase.
- 10) Browse for a Destination you can easily find to save the **.WAR** file to.
- 11) After having found a Destination the Finish button should be clickable, click it.
- 12) Make sure that the *TeradyneCTSDatabase.war* file is in the place you would expect it to be and that it was modified at the time you made the **.WAR** file.

Deploying to the Tomcat Server

Prerequisites

- Completed the Database Installation and CTS System Database Linking sections.
- Have a system which has Apache Tomcat 7.0 installed on it. These directions should work for any Java container Web server but it has been tested exclusively on Apache Tomcat 7.0.
 - Have a User on this system which has privileges to install **.WAR** files.
- Have access to the **.WAR** file made in the CTS System Database Linking chapter.
- A web browser to access the Apache Tomcat Manager App.

Steps

- 1) Log in to the Apache Tomcat Manager App using your browser of choice and the User with install privileges.
- 2) Go to the bottom of the screen under Deploy a Web app and press the Browse button.
- 3) Navigate to the **.WAR** file from the CTS System Database Linking chapter.
- 4) Press the Deploy button to deploy it.
- 5) Make sure the TeradyneCTSDatabase is green, deployed, and is running.

Creating the First User

Prerequisites

- Complete the first three sections (Database Installation, CTS System Database Linking, and Deploying to the Tomcat Server)

Steps

- 1) Navigate to the CTS System. This should be the link to the Tomcat Server followed by “/TeradyneCTSDatabase”. (Example: *tomcat.teradyne.com:8080/TeradyneCTSDatabase/*).
- 2) Log in as the default admin by using the following credentials:
 - Username: `admin`
 - Password: `1234`
- 3) Create a new Administrator User which will be your user by filling out your credentials.
- 4) Log out by pressing the Logout button in the upper right corner of the screen.
- 5) Log in with your new User.
- 6) Remove the admin user.
- 7) Log out again.

7. Appendix B: Maintenance Documentation

WORCESTER POLYTECHNIC INSTITUTE

Maintenance Documentation

Teradyne CTS TPS Database

Version 1.0.2

GFP 1105

Jonathan Marokhovsky

8/11/2012

Table of Contents

LINKING THE SYSTEM TO LDAP.....	33
HOW TO DISCONNECT THE ADMIN CHANGE PASSWORD	34
UNHOOKING THE BUILD HISTORY IN THE HELP SECTION	35
HOW TO CONNECT TO A NEW TYPE OF DATABASE	36
HOW TO MODIFY THE STATE MACHINE	37
HOW TO ADD ATTRIBUTES TO TPS OBJECTS	43
HOW TO MOVE AN EXISTING ORACLE DATABASE.....	45

Linking the System to LDAP

Goal: To allow the CTS Database to utilize the Teradyne login service.

- 1) Configure *com.teradyne.tpsdb.teradyne.TeradyneConnection.java* so that it connects to the Teradyne login system. Currently all the methods in this class are placeholders which don't do anything so replacing the contents of those methods with code that does what the javadocs say the methods should do using the connection to the services will not break anything.
- 2) Configure *com.teradyne.tpsdb.security.SecurityManager.java*'s login method so that instead of checking the internal database for a password field, it is instead checking using the *com.teradyne.tpsdb.teradyne.ITeradyneConnection.java* interface to connect to the Teradyne login system and check with the password there.
- 3) Modify the *WebContent/administration/useradministration/AddUser.jsp* file so that it can better utilize the Teradyne connection.
- 4) Modify *com.teradyne.tpsdb.users.AddUserServlet.java* so that it will take in the new User's information, still add the username to the database, and search the Teradyne system for the given user, possibly redirecting to a different page if the user is not found.
- 5) Make the password field nullable in the Oracle database so that system user passwords are no longer in clear text and no longer supported from the Oracle database (the Teradyne connection should handle this now).
- 6) Remove the Change Your Password ability as that will now be handled through the Teradyne service.

How to disconnect the Admin Change Password

This is how to remove the Change Password functionality when it becomes unnecessary or a security risk.

Requirements

The CTS System is already connected to Teradyne's login service.

Directions

- 1) Open the project for the system.
- 2) Navigate to *WebContent/administration/useradministration/UserManagement.jsp*.
- 3) Find the line which starts with `` and delete from the line above to the line below. At the time of writing this this would be deleting lines **43-45**.
- 4) Find the line which says `<c:if test="${param.form=='changePW'}">` and delete from that line to the end of where it says `</c:if>` two lines under it. At the time of writing this this would be deleting lines **19-21**.

Unhooking the Build History in the Help Section

This is how to remove the Build History section of the Help portion of the system.

Directions

- 1) Open the project for the system.
- 2) Navigate to [WebContent/help/help.jsp](#).
- 3) Find lines **41-43** and remove them. Line **41** should read `<c:when
test="${param.section == 'buildHist' }">`.
- 4) Find lines **29-36** and remove them. Line **30** should also read `<c:when
test="${param.section == 'buildHist' }">`.

How to connect to a new type of database

How to connect to a new type of database

This is how to configure the CTS Database to be able to connect to a new type of database that previously was not supported. For example, the CTS Database currently only supports Oracle databases, this section is if you want the system to be able to connect to a MySQL database or some other database.

Requirements

The database must be able to support

- Triggers
- Inserts, Updates
- Sequences

If any of the above are not supported then the same functionality must be developed in Java.

Interfaces to Instantiate

These are interfaces which must be implemented for a new database connection.

- *com.teradyne.tpsdb.database.IDatabaseConnector.java*
- *com.teradyne.tpsdb.environment.IDatabaseEnvironment.java*
- *com.teradyne.tpsdb.process.IStateProgressionManager.java*
- *com.teradyne.tpsdb.tps.IStateHistoryManager.java*
- *com.teradyne.tpsdb.tps.ITPSManager.java*
- *com.teradyne.tpsdb.users.IUserManager.java*

These are the current implementations of the preceding interfaces which can be used as examples of how to implement the interfaces.

How to Modify the State Machine

- *com.teradyne.tpsdb.database.OracleDatabaseConnector.java*
- *com.teradyne.tpsdb.environment.OracleDatabaseEnvironment.java*
- *com.teradyne.tpsdb.process.OracleStateProgressionManager.java*
- *com.teradyne.tpsdb.tps.OracleStateHistoryManager.java*
- *com.teradyne.tpsdb.tps.OracleTPSManager.java*
- *com.teradyne.tpsdb.users.OracleUserManager.java*

The naming convention is to replace the “I” in the class name which denotes the class as an Interface with the name of the database getting supported. For example, for a MySQL database implementation of the *IDatabaseConnector.java* class one would name the implementing class *MySQLDatabaseConnector.java*.

Creating the Schema

To make a new database schema, convert *createSchema.sql* to a form which the new database will understand.

How to Modify the State Machine

This is how to change the logic of the State machine which TPSs progress down. The state machine is simple and is completely data-driven which means one must only change rows in the database to modify the State table. There is a more in depth explanation of the state machine in the MQP Report.

Tables to Look at / Insert Rows Into

- States

How to Modify the State Machine

- These are the actual states, inserting rows into this table will add points in the state machine
- StateActions
 - These are the connections between states, inserting rows into this table will connect states in the machine
- Actions
 - These are the names of actions as they will show up to Users, inserting rows into this table will give more possible names of actions for the StateActions table to use.
- DeveloperTypes
 - These are the paths which States follow; currently there are only three, Not Applicable, Teradyne Apps, and ASP. Not Applicable is used for paths which both Teradyne Apps and ASP follow.
 - *CAUTION*: this also affects the types of developers as the name implies.

Example case

Say I wanted to add a state in the state machine right before the Closed state named “**Ice Cream Party**” with an Action coming from it and leading it to the Closed state named “**Recover from Stomach Ache**”. I would:

1. First I would focus on making the new State.
 - a. Query the database to find the largest `state_ID`. The code I would use (first result is what you care about):

How to Modify the State Machine

i. `SELECT state_ID FROM States ORDER BY state_ID
DESC;`

- b. Say the largest ID in this case is 21. Adding 1 to this will give you your new `state_ID`. So the new `state_ID` will be 22.
- c. Find out the `state_path` to use by running the following query on the database and then choosing the `dev_ID` which corresponds to the desired path.

i. `SELECT dev_ID, dev_name FROM DeveloperTypes;`

- d. In this case we are on the main path so the `state_path` will be 0.
- e. Now we can build our `INSERT` statement for the new state:

i. `INSERT INTO States (state_ID, state_name,
state_desc, state_path) VALUES(22, 'Ice Cream
Party', 'Throw an ice cream party to celebrate
closing a TPS', 0);`

- f. Make a new `.SQL` file and put this `INSERT` statement into it. I would name the `.SQL` file `newState.sql`. Save this file and set it aside.

2. Now make the new Action.

- a. Find the largest `action_ID`.

i. `SELECT action_ID FROM Actions ORDER BY action_ID
DESC;`

- b. Say in this case the highest `action_ID` is 22. Add 1 to this to make it your new `action_ID`. So the new `action_ID` will be 23.
- c. Build the `INSERT` statement for the new Action:

How to Modify the State Machine

i. `INSERT INTO Actions (action_ID, action_name)
VALUES(23, 'Recover from stomach ache');`

d. Add this INSERT statement to *newState.sql*.

3. Now make the new StateAction.

a. Find out the ID of the State which will be on the other end of the Action you are implementing.

i. `SELECT state_ID, state_name FROM States`

b. Since we are going to the Closed state the state_ID for that is 21.

c. Knowing this we can now build the StateAction INSERT statement with this information:

i. `Current_state = 22, new_state=21, event_value = 23.`

ii. `INSERT INTO StateActions (current_state,
new_state, event_table, event_col_name,
event_value_key, event_value) VALUES(22, 21,
'Actions', 'Action_name', 'Action_ID', 23);`

d. Add this INSERT statement to *newState.sql*. **Make sure this statement is under the State INSERT statement and the Action INSERT statement.**

4. Make the Revert StateAction to go from our new state to the state before it by following the last step and having the event_value be 0 which is the ID for the Revert action.

a. Make the revert StateAction should end up looking like this:

i. `INSERT INTO StateActions (current_state,
new_state, event_table, event_col_name,`

How to Modify the State Machine

```
event_value_key, event_value) VALUES(22, 20,  
'Actions', 'Action_name', 'Action_ID', 0);
```

- b. Add this INSERT statement to *newState.sql*. **Make sure this statement is under the State INSERT statement and the Action INSERT statement.**
5. Update the StateActions which are displaced by adding this new State in the flow.
 - a. Look up the old StateAction which used to connect the State before our new State to the State after our new state. We already know its ID from the last step. We are doing this to find out the action ID to use because StateActions are unique by a combination of their `current_state`, `event_table`, and `event_value` columns.
 - i.

```
SELECT * FROM StateActions WHERE current_state =  
20 AND new_state = 21;
```
 - b. The columns values in this case are `event_table = 'Actions'` and `event_value = 22`
 - c. Take all the information from that to make the UPDATE statement.
 - i.

```
UPDATE StateActions SET new_state = 22 WHERE  
current_state = 20 AND event_table = 'Actions'  
AND event_value=22;
```
 - d. Add this UPDATE statement to *newState.sql*. **Make sure this statement is under the State INSERT statement and the Action INSERT statement.**
6. In most cases the last step would have to be repeated for the Revert StateAction belonging to the State after our Ice Cream state but since that State is the Closed state in this case That is unnecessary.

How to Modify the State Machine

7. Run the *newState.sql* script on the database now.
8. Add the contents of the *newState.sql* script to the bottom of the *createSchema.sql* script to make sure when creating a new database for this system these changes are also included.

How to Add Attributes to TPS Objects

This is how to change what fields a TPS has. For example, if there is now an attribute for a sub-owner for TPSs this section is where one should start.

Places to Look to Change

- Database Level
 - In *createSchema.sql*
 - TPSs table (search for “*CREATE TABLE TPSS*”)
 - This is where all TPS information is stored
 - The TPS Update Trigger (search for “*trg_before_update_tps*”)
 - This controls how a TPS acts when it is updated
- Java Level
 - *com.teradyne.tpsdb.tps.ITPS.java*
 - The interface controlling what information can be acquired from a TPS object in Java
 - *com.teradyne.tpsdb.tps.TPS.java*
 - The instantiated class which controls how the information is actually controlled in Java
 - *com.teradyne.tpsdb.tps.ITPSManager.java*
 - The interface which handles how TPS information is acquired from whatever database the system is connected to.
 - *com.teradyne.tpsdb.tps.OracleTPSManager.java*

How to Add Attributes to TPS Objects

- How the system actually interprets the TPS information it gets from an Oracle database.
- *com.teradyne.tpsdb.tps.TPSBean.java*
 - This is used to make it easier for the information in a TPS to be interpreted by the view.
 - *com.teradyne.tpsdb.tps.TPSBeanFactory.java*
 - This controls the conversion from the TPS object to the TPSBean object.
 - Any attributes added to the TPSBean should be added to the convertor in this class
- *com.teradyne.tpsdb.tps.TPSFactory.java*
 - This abstracts to another level how information is gathered from databases.
- Any other class in the *com.teradyne.tpsdb.tps* package
- JSP Level (View)
 - *WebContent/management/searchResults.jsp*,
WebContent/engineering/viewMyTPS.jsp, and
WebContent/engineering/superViewMyTPS.jsp
 - These are the views where a list of TPSs and some important information
 - *WebContent/engineering/wholeTPS.jsp*
 - This is the screen which will show all information for a given TPS
 - Most other files in the *WebContent/engineering/* directory and the *adminTPSFunctions* sub-directory

How to Move an existing Oracle Database

- This is assuming that the Oracle Database is 10g and above, any database older than that should use the export/import commands
- In the command line use the expdp/impdp commands
- See www.orafaq.com/wiki/Data_Pump for actual directions on how to do this.

8. Appendix C: User Documentation

WORCESTER POLYTECHNIC INSTITUTE

User Documentation

Teradyne CTS TPS Database

Version: 1.0.2

GFP 1105

Jonathan Marokhovsky

8/11/2012

Table of Contents

ALL FORMS	48
THE RESET BUTTON.....	48
THE CLEAR BUTTON	48
CANCEL BUTTONS.....	48
<i>User Administration</i>	48
<i>Administrator TPS Functions</i>	48
<i>TPS Management</i>	49
ADMINISTRATOR FUNCTIONS	50
<i>Precondition to All Administrator Functions</i>	50
<i>Precondition to All Administrator TPS Management Functions</i>	50
ADD USER	51
REMOVE USER.....	53
CHANGE USER PRIVILEGE	54
CHANGE A USER'S PASSWORD	55
ADMINISTRATOR TPS MANAGEMENT VIEW ALL TPSS IN THE SYSTEM.....	56
ADMINISTRATOR TPS MANAGEMENT TPS EDIT	56
ADMINISTRATOR TPS MANAGEMENT VIEW ALL OF THE INFORMATION PERTAINING TO A TPS.....	57
ADMINISTRATOR TPS MANAGEMENT CHANGE A TPS'S STATE TO ANY STATE.....	58
ADMINISTRATOR TPS MANAGEMENT CHANGE A TPS'S OWNER	60
BASIC USER FUNCTIONS	62
<i>Preconditions to All Basic User Functions</i>	62
CREATE A NEW TPS	63
VIEW ALL TPSS OWNED BY THE LOGGED IN USER	65
VIEW ALL OF THE INFORMATION PERTAINING TO A TPS	66
EDIT A TPS	67
CHANGE A TPS'S STATE.....	69
VIEW-ONLY USER FUNCTIONS	71
QUERYING FOR TPSS	71

All Forms

The following buttons are present on all forms in the system.

The Reset Button

The Reset button, if present on a form, will set all fields on the form back to what was present upon loading of the page. For example, if a date field says 08/15/2013 and you change it to say 09/23/3013 pressing the Reset button will make it say once again 08/15/2013.

The Clear Button

The Clear button, if present on a form, will set all fields on the form back to a blank. This is typically present on a form which starts out with all fields completely blank.

Cancel Buttons

User Administration

Pressing the Cancel button on any of the forms under the Administration tab results in returning to the main User Management screen without changing anything in the database. This is the same result as pressing the “Back to User Management” link at the top of the screen, the User Management button on the left side of the screen, or the Administration tab again.

Administrator TPS Functions

Pressing the Cancel button on any of the forms that are classified as Administrator TPS Management will send the User back to the Admin TPS View screen. This is the same result as pressing the Admin TPS View button.

All Forms

TPS Management

Pressing the Cancel button on any form from the TPS Management screen other than those that are Administrator TPS Functions will return the User to the View My TPSs screen.

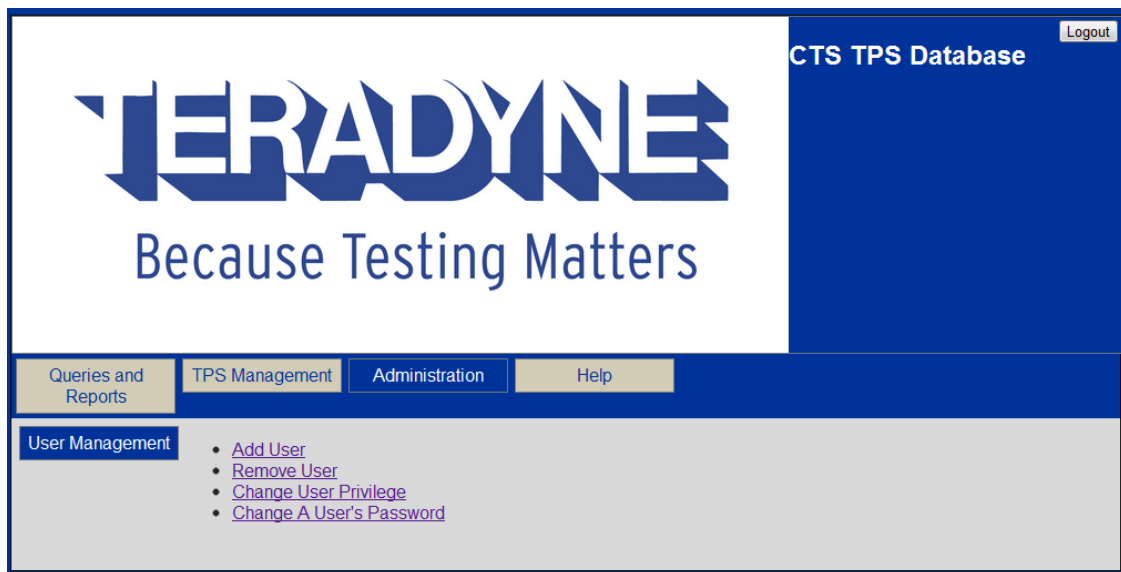
Administrator Functions

An administrator is a User who has the responsibility of maintaining the CTS system. These responsibilities include User Maintenance and correcting mistakes in TPSs by being able to move any TPS to any state. Because of their responsibilities in maintenance of the system, Administrators also have access to the same functions as Basic Users and View-Only Users.

Note: The Administrator TPS Management Functions are functions which only an Administrator has access to help maintain TPSs in the system.

Precondition to All Administrator Functions

- Log in as a User with administrator privileges. This should place the User automatically at the Administration screen.



Precondition to All Administrator TPS Management Functions

Administrator Functions

- 1) Navigate to the TPS Management screen.

CTS TPS Database

Logout

TERADYNE
Because Testing Matters

Queries and Reports | **TPS Management** | Administration | Help

View My TPSs	TPS Number	Cert Pack Rev	Cert Pack Iter	Cert Data Rev	Cert Data Iter	State	Release Date	Actions
Create a TPS	111-111-14	AA	5			Created	08/07/2012	Edit Change State
Admin TPS View								

- 2) Press the Admin TPS View button. This should place the User at the View All TPSs screen.

CTS TPS Database

Logout

TERADYNE
Because Testing Matters

Queries and Reports | TPS Management | Administration | Help

View My TPSs | Admin TPS Abilities

View My TPSs	TPS Number	Cert Pack Rev	Cert Pack Iter	Cert Data Rev	Cert Data Iter	State	Owner	Actions
Create a TPS	111-111-14	AA	5			Created	The admin	Edit Change State Change Owner
Admin TPS View	141-414-14			J	14	Created	John Doe	Edit Change State Change Owner

Add User

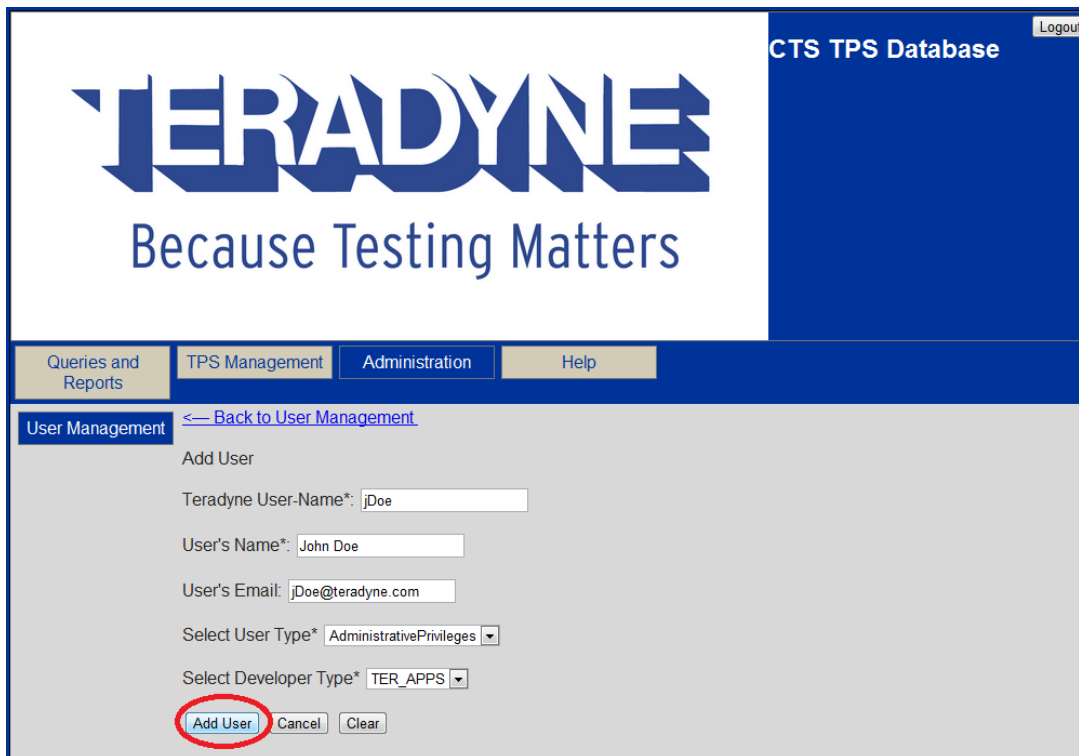
Goal: To add a new User to the Teradyne CTS Database System. This new User must already be an employee of Teradyne.

Administrator Functions

How to use it:



- 1) Press the Add User link.
- 2) Fill out the new User's information.



- 3) Press the Add User button.

Administrator Functions

Remove User

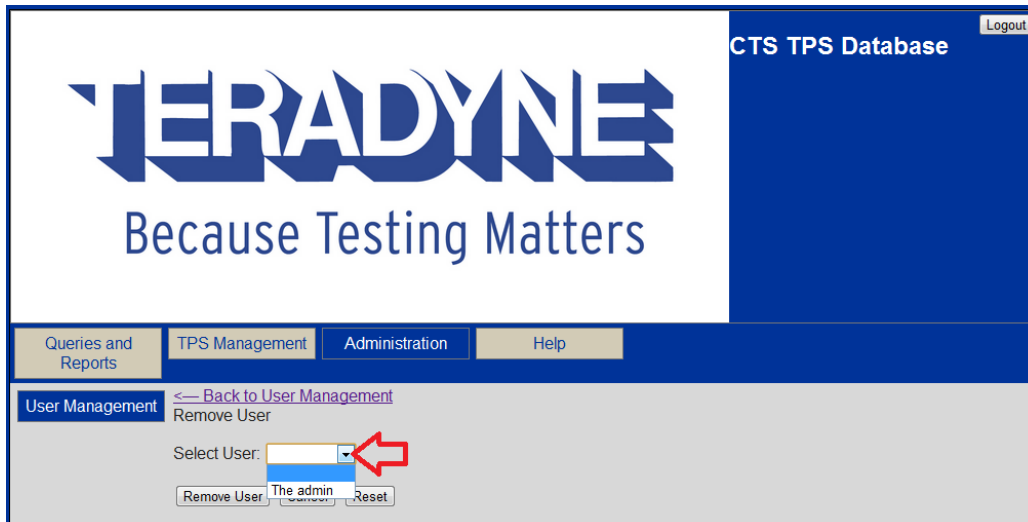
Goal: To remove an existing User from the Teradyne CTS Database System.

Warning: This cannot be undone from the application and requires someone who manages the SQL database to undo this.

How to use it:



1) Press the Remove User link.



2) Select the User to be removed from the Select User dropdown menu.

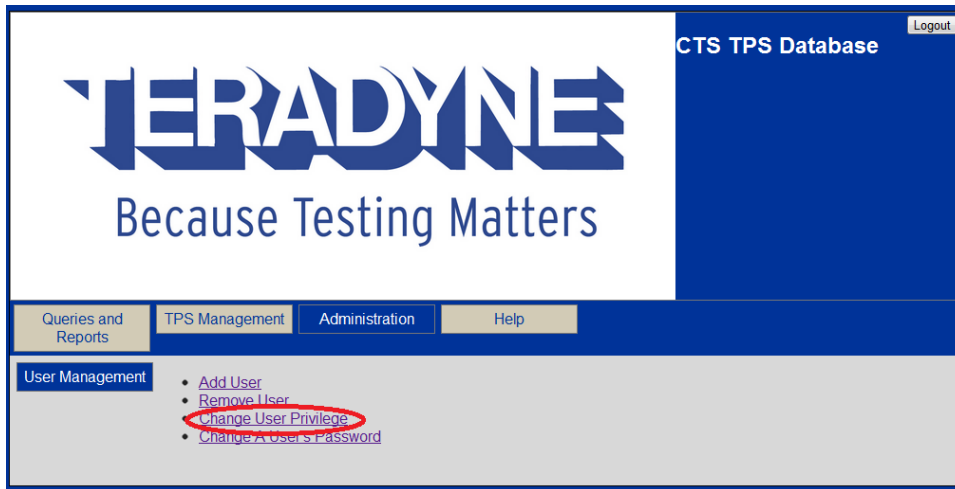
Administrator Functions

3) Press the Remove User button. This will immediately remove the User from the system.

Change User Privilege

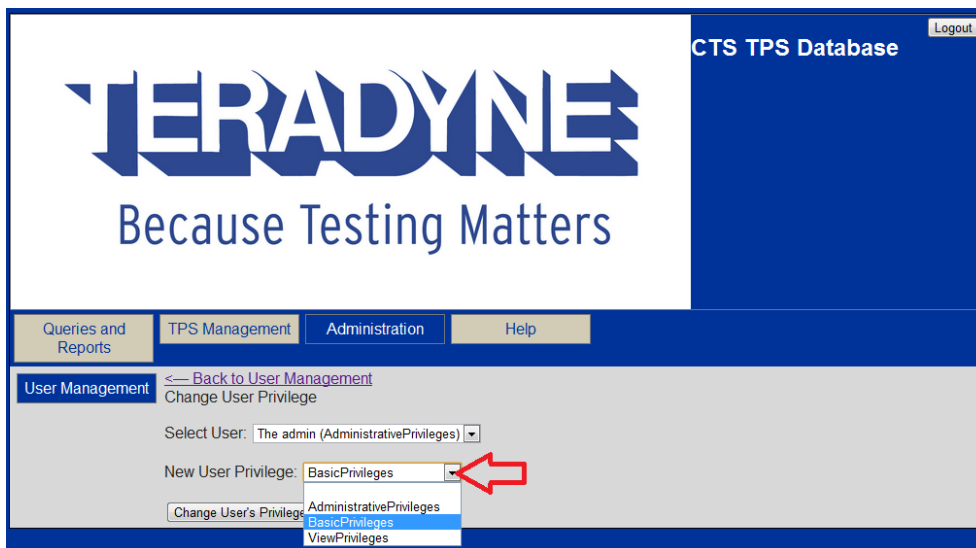
Goal: To change an existing User's privileges.

How to use it:



1) Press the Change User Privilege link.

2) Select the User who's privilege will be changed from the "Select User" dropdown menu.



3) Select the new User Privilege from the "New User Privilege" dropdown menu.

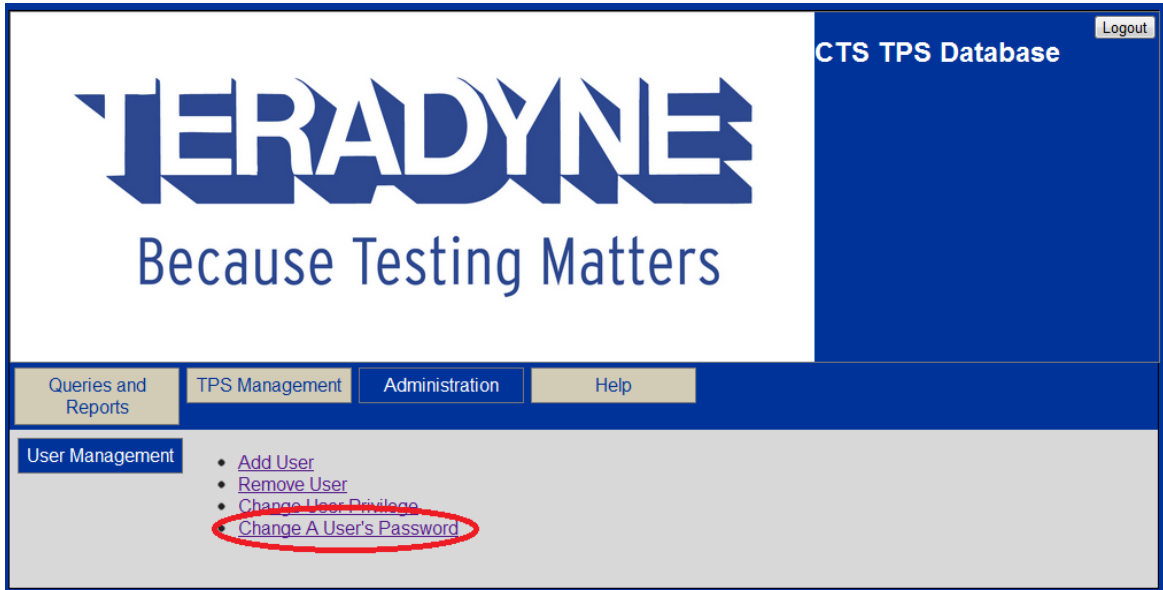
4) Press the Change User's Privilege button.

Change A User's Password

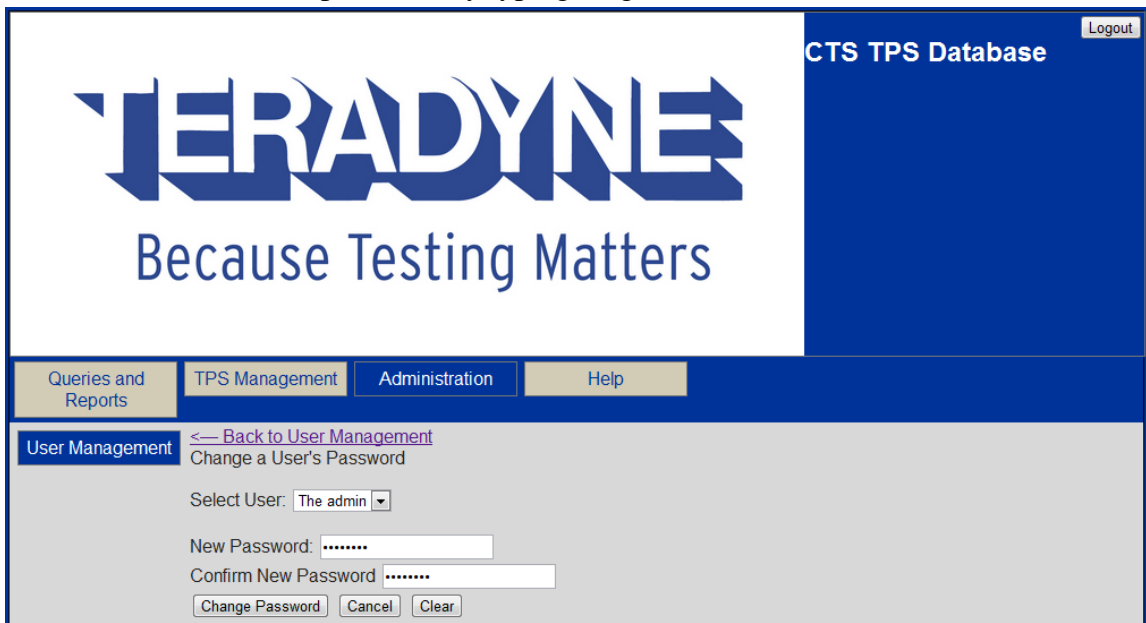
Goal: To change a User's password.

How to use it:

1. Press the Change A User's Password link.



2. Select the User who's password is to be changed.
3. Type in the User's new password.
4. Confirm the User's new password by typing it again in the confirmation field.



Administrator Functions

Administrator TPS Management | View All TPSs in the System

Goal: To view all TPSs in the system regardless of owner.

How to use it:

- 1) Follow the steps under the **PRECONDITION TO ALL ADMINISTRATOR TPS MANAGEMENT FUNCTIONS** section; this will land the User on the Admin TPS View page.
 - This will show all TPSs in the system, for all users.

Administrator TPS Management | TPS Edit

Goal: To edit all fields in a TPS.

How to use it:

- 1) Find the TPS to be edited in the list of TPSs.
- 2) Press the Edit button next to the TPS's information under the Actions column.

TPS Number	Cert Pack Rev	Cert Pack Iter	Cert Data Rev	Cert Data Iter	State	Owner	Actions
111-111-14	AA	5			Created	The admin	Edit Change State Change Owner
141-414-14			J	14	Created	John Doe	Edit Change State Change Owner

- 3) Change whatever information is necessary.
- 4) Add a note if necessary.

Administrator Functions

5) Press Submit to submit the edits on the TPS.

Queries and Reports | **TPS Management** | Administration | Help

View My TPSs
Create a TPS
Admin TPS View

Edit TPS

Report Information

TPS Number: 141 - 414 - 14

Release Date (MM/DD/YYYY): 11/14/2012

Numbers

Cert Pack Rev Num:

Cert Pack Iter Num:

Cert Data Rev Num: J

Cert Data Iter Num: 14

Notes

This is a new note.

Administrator TPS Management | View All of the Information Pertaining to a TPS

Goal: To view information on any TPS which exists in the system.

How to use it:

1) Find the TPS in the list of TPSs.

Administrator Functions

- 2) Press on the TPS's TPS Number.

The screenshot shows the CTS TPS Database interface. At the top left is the TERADYNE logo with the tagline "Because Testing Matters". The top right corner has "CTS TPS Database" and a "Logout" link. Below the logo is a navigation bar with tabs for "Queries and Reports", "TPS Management", "Administration", and "Help". The main content area has a sidebar with "View My TPSs", "Create a TPS", and "Admin TPS View". The main table is titled "Admin TPS Abilities" and contains the following data:

TPS Number	Cert Pack Rev	Cert Pack Iter	Cert Data Rev	Cert Data Iter	State	Owner	Actions
111-111-14	AA	5			Created	The admin	<input type="button" value="Edit"/> <input type="button" value="Change State"/> <input type="button" value="Change Owner"/>
141-414-14			J	14	Created	John Doe	<input type="button" value="Edit"/> <input type="button" value="Change State"/> <input type="button" value="Change Owner"/>

- 3) The information will show up in a new window or tab depending on browser settings.

TPS Number	Cert Data Rev Number	Cert Data Iter Number	Cert Pack Rev Number	Cert Pack Iter Number	Current State	Owner	Creator	Date Created	Last Modified On	Release Date	Closed?	Most Recent Note		
												Note	Author	Creation Time
141-414-14	J	14			Created	John Doe		08/08/2012	08/08/2012	11/14/2012	No	This is my TPS.	The admin	2012-08-08 06:01:40.386101
												mail	The admin	2012-08-08 06:10:20.930985

- This is so that a User can continue to work in the system while keeping the information up.

- 4) Close out of the window when done using the information.

Administrator TPS Management | Change a TPS's State to Any State

Goal: To move a TPS to any possible state.

How to use it:

- 1) Find the TPS to progress in the list of TPSs.

Administrator Functions

- 2) Press the Change State button next to the TPS's information under the Actions column.

The screenshot shows the TERADYNE CTS TPS Database interface. The header includes the TERADYNE logo and the slogan "Because Testing Matters". The page title is "CTS TPS Database" with a "Logout" link. The navigation menu includes "Queries and Reports", "TPS Management", "Administration", and "Help". The main content area is titled "Admin TPS Abilities" and contains a table of TPS records. The "Change State" button for the first record is highlighted with a red circle.

TPS Number	Cert Pack Rev	Cert Pack Iter	Cert Data Rev	Cert Data Iter	State	Owner	Actions
111-111-14	AA	5			Created	The admin	<input type="button" value="Edit"/> <input type="button" value="Change State"/> <input type="button" value="Change Owner"/>
141-414-14			J	14	Created	John Doe	<input type="button" value="Edit"/> <input type="button" value="Change State"/> <input type="button" value="Change Owner"/>

- 3) Select the correct action to be done to the TPS from the "Choose the Action" dropdown list.


The screenshot shows the TERADYNE CTS TPS Database interface with a dropdown menu open for selecting an action for a specific TPS record. The dropdown menu lists various actions such as "Created", "Cert Pack Checked In", "CQ State Eng Approved", etc.

TPS Number: 111-111-14

- Created
- Cert Pack Checked In
- CQ State Eng Approved
- Cert Data Audited
- Cert Pack Built
- Cert Pack Audited
- Checked Into VSS
- Cert Pack Checked In
- Artwork Requested
- ISO Created
- ISO Audited
- CQ State CS_Approved
- ECN Initiated
- Master CD & ISO Verified
- Master CD in SDC
- CDs Sent to Customers
- TPS Index Updated
- Closed

Administrator Functions

- 4) Press the Submit button to submit the state change.



The screenshot displays the TERADYNE CTS TPS Database interface. The top left features the TERADYNE logo and the tagline "Because Testing Matters". The top right shows "CTS TPS Database" and a "Logout" button. A navigation bar includes "Queries and Reports", "TPS Management" (highlighted), "Administration", and "Help". Below this, there are buttons for "View My TPSs", "Create a TPS", and "Admin TPS View". The "Admin TPS View" section includes a "TPS Number: 111-111-14" field, a dropdown menu, and three buttons: "Select" (circled in red), "Cancel", and "Clear".

Administrator TPS Management | Change a TPS's Owner

Goal: To send a TPS to another owner without changing the TPS's state.

How to use it:

- 1) Find the TPS to progress in the list of TPSs.

Administrator Functions

- 2) Press the Change Owner button next to the TPS's information under the Actions column.

The screenshot shows the TERADYNE CTS TPS Database interface. The header includes the TERADYNE logo and the slogan "Because Testing Matters". The navigation menu includes "Queries and Reports", "TPS Management", "Administration", and "Help". The main content area is titled "Admin TPS Abilities" and contains a table with the following data:

TPS Number	Cert Pack Rev	Cert Pack Iter	Cert Data Rev	Cert Data Iter	State	Owner	Actions
111-111-14	AA	5			Created	The admin	<input type="button" value="Edit"/> <input type="button" value="Change State"/> <input type="button" value="Change Owner"/>
141-414-14			J	14	Created	John Doe	<input type="button" value="Edit"/> <input type="button" value="Change State"/> <input type="button" value="Change Owner"/>

The "Change Owner" button for the first row is circled in red.

- 3) Select the new owner of the TPS from the "The New Owner Will Be" dropdown list.

The screenshot shows the TERADYNE CTS TPS Database interface with the "Change the Owner" dialog box open. The dialog box displays the TPS Number: 111-111-14 and the "The New Owner Will Be:" dropdown menu. The dropdown menu is open, showing the following options:

- John Doe
- The admin
- John Doe

The "Change the Owner" button is highlighted in blue.

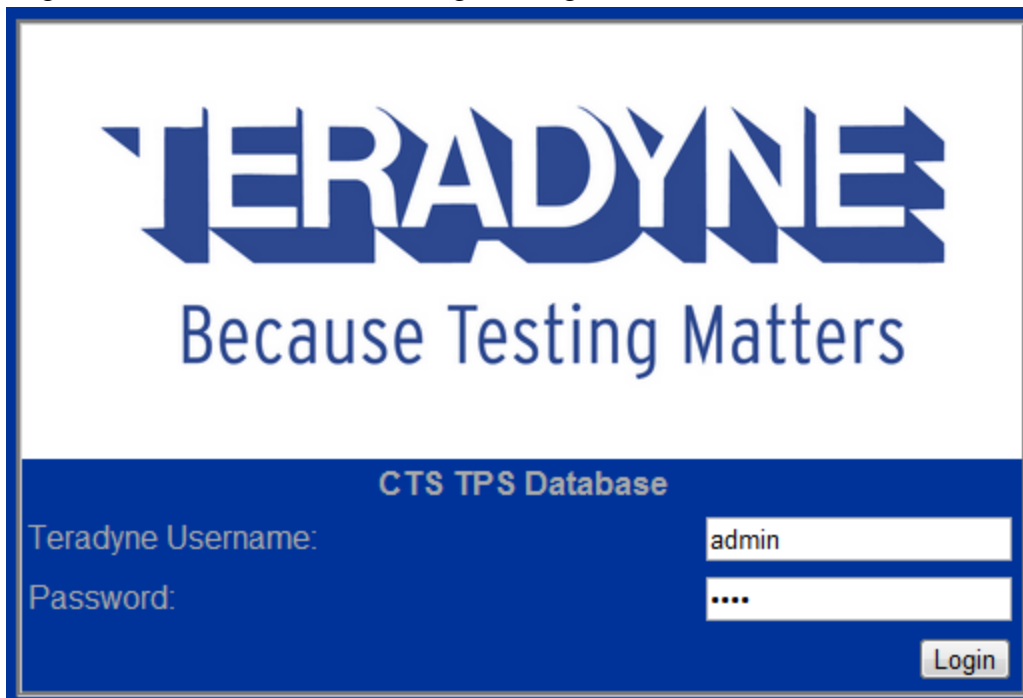
- 4) Press the Submit button to submit the state change.

Basic User Functions

A Basic User is the average user of the CTS TPS Database. This privilege level is meant for those with the responsibility of creating new TPSs and progressing them from creation to completion. Basic Users also have access to all View-Only User functions.

Preconditions to All Basic User Functions

- 1) Log in as a User with Basic Privileges or higher.



The screenshot shows the Teradyne login interface. At the top, the Teradyne logo is displayed in a stylized blue font, with the tagline "Because Testing Matters" below it. The interface is titled "CTS TPS Database" in white text on a blue background. Below the title, there are two input fields: "Teradyne Username:" with the value "admin" and "Password:" with four dots representing a masked password. A "Login" button is located at the bottom right of the form.

Basic User Functions

- 2) If not automatically on the TPS Management tab, click on the tab to navigate to the correct screen.

TERADYNE
Because Testing Matters

CTS TPS Database
Logout

Queries and Reports | **TPS Management** | Administration | Help

View My TPSs	TPS Number	Cert Pack Rev	Cert Pack Iter	Cert Data Rev	Cert Data Iter	State	Release Date	Actions
Create a TPS	111-111-14	AA	5			Created	08/07/2012	Edit Change State
Admin TPS View								

Create a new TPS

Goal: To Create a new TPS and put it in the system.

Basic User Functions

How to use it:

- 1) Press the “Create a TPS” button to go to the Create TPS screen.

CTS TPS Database

Logout

TERADYNE
Because Testing Matters

Queries and Reports | TPS Management | Administration | Help

View My TPSs	TPS Number	Cert Pack Rev	Cert Pack Iter	Cert Data Rev	Cert Data Iter	State	Release Date	Actions
Create a TPS								Edit
Admin TPS View	111-111-14	AA	5			Created	08/07/2012	Change State

- 2) Fill in all necessary information for the TPS.

Basic User Functions

3) Any additional information can go in the Notes section.

The screenshot shows a web application interface for creating a TPS. At the top, there is a navigation bar with four tabs: 'Queries and Reports', 'TPS Management' (which is active), 'Administration', and 'Help'. On the left side, there is a sidebar with three buttons: 'View My TPSs', 'Create a TPS' (which is highlighted in blue), and 'Admin TPS View'. The main content area is titled 'Create TPS' and is divided into three sections: 'Report Information', 'Numbers', and 'Notes'. The 'Report Information' section contains three fields: 'Developer Type' (a dropdown menu with 'ASP' selected), 'First Responsible Developer' (a dropdown menu with 'The admin' selected), and 'Release Date (MM/DD/YYYY)' (a text input field with '08/07/2012' entered). The 'Numbers' section contains five fields: 'TPS Number' (a text input field with '111' entered), a hyphen, another 'TPS Number' (a text input field with '111' entered), a hyphen, and a third 'TPS Number' (a text input field with '14' entered). Below these are four more fields: 'Certification Pack Revision Letter(s)' (a text input field with 'AA' entered), 'Certification Pack Iteration Number' (a text input field with '5' entered), 'Certification Data Revision Letter(s)' (an empty text input field), and 'Certification Data Iteration Number' (an empty text input field). The 'Notes' section is a large text area containing the text 'Hello World.'. At the bottom of the form, there are three buttons: 'Create TPS Report', 'Clear', and 'Cancel'.

4) Press Create TPS Report to create the TPS.

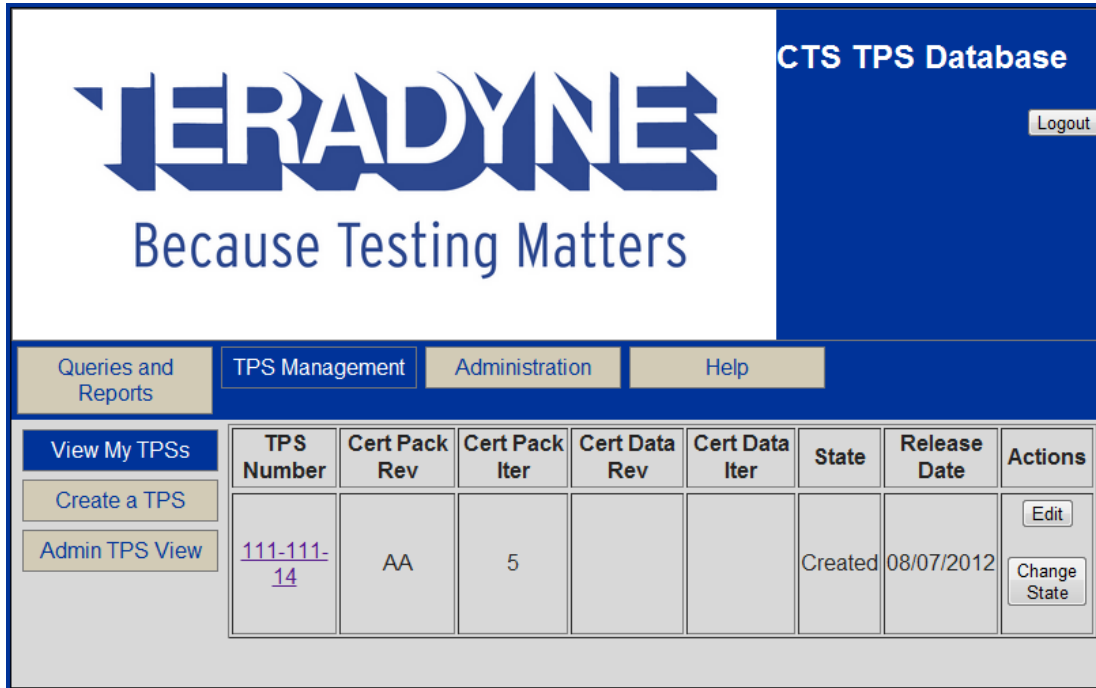
View All TPSs Owned by the Logged in User

Goal: To view all TPSs which the currently logged in User owns.

Basic User Functions

How to use it:

- 1) The list of TPSs there are all TPSs owned by the currently logged in User.



The screenshot displays the TERADYNE CTS TPS Database interface. The header includes the TERADYNE logo and the slogan "Because Testing Matters". The page title is "CTS TPS Database" with a "Logout" button. A navigation bar contains "Queries and Reports", "TPS Management" (selected), "Administration", and "Help". On the left, there are buttons for "View My TPSs", "Create a TPS", and "Admin TPS View". The main content is a table with the following data:

TPS Number	Cert Pack Rev	Cert Pack Iter	Cert Data Rev	Cert Data Iter	State	Release Date	Actions
111-111-14	AA	5			Created	08/07/2012	<input type="button" value="Edit"/> <input type="button" value="Change State"/>

View All of the Information Pertaining to a TPS

Goal: To view information on a TPS which already exists in the system and is owned by the User.

How to use it:

- 1) Find the TPS in the list of owned TPSs.

- 2) Press on the TPS's TPS Number.

- 3) The information will show up in a new window or tab depending on browser settings.

TPS Number	Cert Data Rev Number	Cert Data Iter Number	Cert Pack Rev Number	Cert Pack Iter Number	Current State	Owner	Creator	Date Created	Last Modified On	Release Date	Closed?	Most Recent Note		
141-414-14	J	14			Created	John Doe		08/08/2012	08/08/2012	11/14/2012	No	Note	Author	Creation Time
												This is my TPS.	The admin	2012-08-08 06:01:40.386101
												null	The admin	2012-08-08 06:10:20.930985

- a) This is so that one can continue to work in the system while keeping the information up.
 4) Close out of the window when done using the information.

Edit a TPS

Goal: To edit the information of a TPS which already exists in the system and is owned by the User.

How to use it:

- 1) Find the TPS to be edited in the list of owned TPSs.

Basic User Functions

- 2) Press the Edit button next to the TPS's information under the Actions column.

The screenshot displays the TERADYNE CTS TPS Database interface. At the top left is the TERADYNE logo with the tagline "Because Testing Matters". The top right corner shows "CTS TPS Database" and a "Logout" button. Below the header is a navigation bar with tabs for "Queries and Reports", "TPS Management", "Administration", and "Help". The main content area features a sidebar with "View My TPSs", "Create a TPS", and "Admin TPS View" buttons. The central table lists TPS information with columns for TPS Number, Cert Pack Rev, Cert Pack Iter, Cert Data Rev, Cert Data Iter, State, and Release Date. The "Actions" column for the first row contains an "Edit" button (circled in red) and a "Change State" button. The data row shows TPS Number 111-111-14, Cert Pack Rev AA, Cert Pack Iter 5, State Created, and Release Date 08/07/2012.

TPS Number	Cert Pack Rev	Cert Pack Iter	Cert Data Rev	Cert Data Iter	State	Release Date	Actions
111-111-14	AA	5			Created	08/07/2012	Edit Change State

- 3) Change whatever information that is necessary.

Basic User Functions

- 4) Add a note if necessary.

[View My TPSs](#) **Edit TPS** [Create a TPS](#)

Report Information
TPS Number: 141-414-14
Release Date (MM/DD/YYYY): 11/14/2012

Numbers
Cert Pack Rev Num:
Cert Pack Iter Num:
Cert Data Rev Num: J
Cert Data Iter Num: 14

Notes

Note	Author	Creation Time
This is my TPS.	The admin	2012-08-08 06:01:40.386101
null	The admin	2012-08-08 06:10:20.930985

This TPS's State History

From State	To State	From User	Change Date
Created but Without Developer	Created	The admin	08/08/2012

- 5) Press Submit to submit the edits on the TPS.

Change a TPS's state

Goal: To move a TPS which the logged in User owns to a new state. Changing a TPS's state requires that a new Owner be assigned to the TPS as well.

How to use it:

- 1) Find the TPS to progress in the list of owned TPSs.

Basic User Functions

- 2) Press the Change State button next to the TPS's information under the Actions column.

The screenshot shows the TERADYNE CTS TPS Database interface. The header includes the TERADYNE logo and the slogan "Because Testing Matters". The page title is "CTS TPS Database" with a "Logout" button. The navigation menu includes "Queries and Reports", "TPS Management", and "Help". The main content area has a "View My TPSs" button and a "Create a TPS" button. Below these is a table with the following columns: TPS Number, Cert Pack Rev, Cert Pack Iter, Cert Data Rev, Cert Data Iter, State, Release Date, and Actions. The table contains one row with the following data: TPS Number: 141-414-14, Cert Pack Rev: (empty), Cert Pack Iter: (empty), Cert Data Rev: J, Cert Data Iter: 14, State: Created, Release Date: 11/14/2012. The Actions column for this row contains an "Edit" button and a "Change State" button, which is circled in red.

TPS Number	Cert Pack Rev	Cert Pack Iter	Cert Data Rev	Cert Data Iter	State	Release Date	Actions
141-414-14			J	14	Created	11/14/2012	<input type="button" value="Edit"/> <input type="button" value="Change State"/>

- 3) Select the correct action to be done to the TPS from the "Choose the Action" dropdown list.
- 4) Select the new owner of the TPS from the "The New Owner Will Be" dropdown list.

The screenshot shows the "Change State" form in the TERADYNE CTS TPS Database interface. The navigation menu includes "Queries and Reports", "TPS Management", and "Help". The form has a "View My TPSs" button and a "Create a TPS" button. The form displays the TPS Number: 141-414-14. The "Choose the Action:" dropdown menu is set to "Built Cert Pack & Checked In". The "The New Owner Will Be:" dropdown menu is open, showing a list of options: "The admin" and "John Doe". The form also includes "Submit", "Cancel", and "Clear" buttons.

TPS Number: 141-414-14

Choose the Action: Built Cert Pack & Checked In

The New Owner Will Be:
The admin
John Doe

- 5) Press the Submit button to submit the state change.

View-Only User Functions

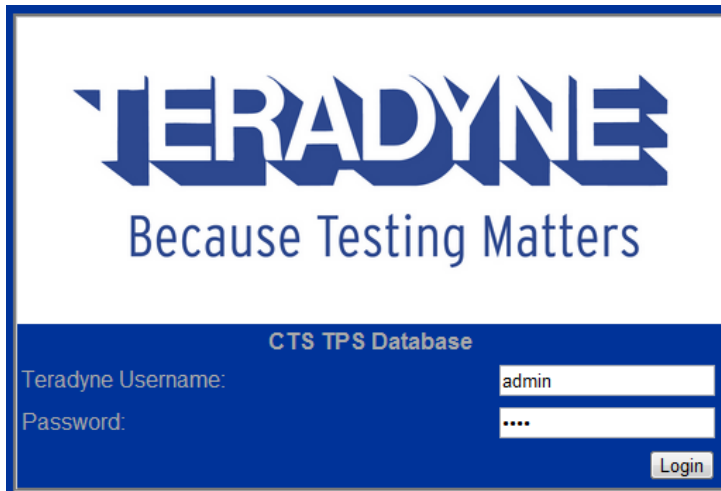
A View-Only User only has access to see information in the CTS TPS Database. This User privilege does not have any way of modifying information. This privilege level is good for those who do not need access to modify or progress any TPSs, but need to gather information on how TPSs are progressing and see patterns.

Querying for TPSs

Goal: To find information on a given TPS.

How to use it:

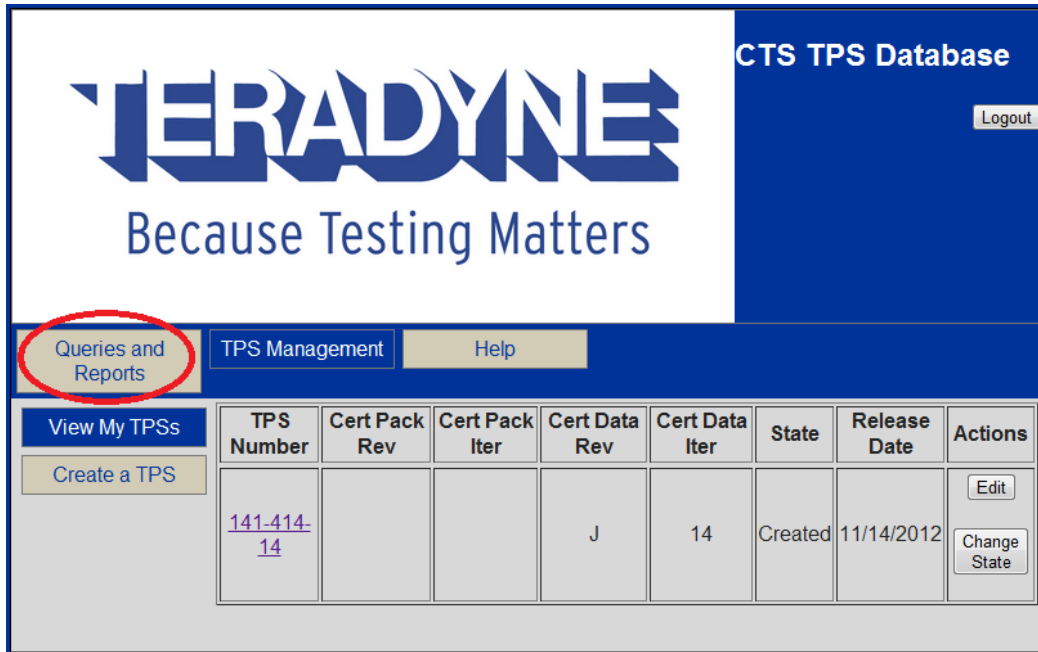
- 1) Log in as a User with at least view-only privileges.



The image shows a login interface for the Teradyne CTS TPS Database. At the top, the Teradyne logo is displayed in blue, with the tagline "Because Testing Matters" below it. The page title "CTS TPS Database" is centered above the login fields. The login form consists of two input fields: "Teradyne Username:" with the value "admin" and "Password:" with masked characters "****". A "Login" button is located at the bottom right of the form.

View-Only User Functions

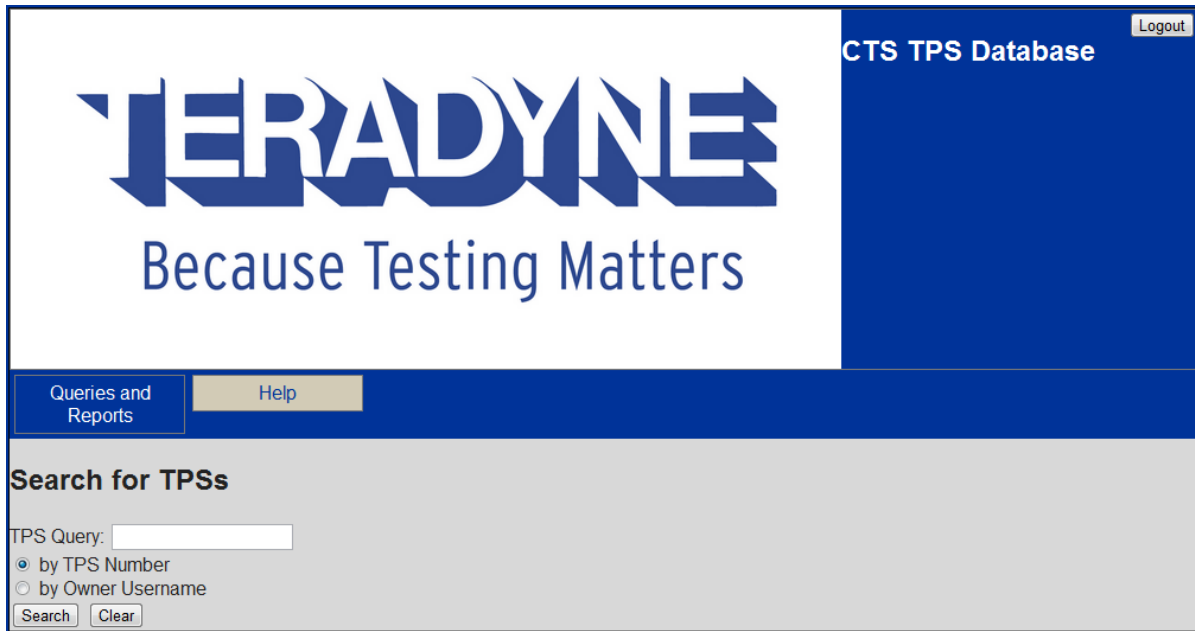
- 2) If not already on the Queries and Reports tab, press the tab to go to the appropriate section.



The screenshot shows the TERADYNE CTS TPS Database interface. The header includes the TERADYNE logo and the slogan "Because Testing Matters". The page title is "CTS TPS Database" with a "Logout" button. A navigation bar contains three tabs: "Queries and Reports" (highlighted with a red circle), "TPS Management", and "Help". Below the navigation bar, there are two buttons: "View My TPSs" and "Create a TPS". A table displays the following data:

TPS Number	Cert Pack Rev	Cert Pack Iter	Cert Data Rev	Cert Data Iter	State	Release Date	Actions
141-414-14			J	14	Created	11/14/2012	<input type="button" value="Edit"/> <input type="button" value="Change State"/>

- 3) Type in the TPS number of the desired TPS.
4) Make sure the TPS Number radio button is selected.



The screenshot shows the TERADYNE CTS TPS Database interface with the "Queries and Reports" tab selected. The header includes the TERADYNE logo and the slogan "Because Testing Matters". The page title is "CTS TPS Database" with a "Logout" button. A navigation bar contains two tabs: "Queries and Reports" and "Help". Below the navigation bar, there is a "Search for TPSs" section with a text input field for "TPS Query:". Below the input field, there are two radio buttons: "by TPS Number" (selected) and "by Owner Username". At the bottom of the search section, there are "Search" and "Clear" buttons.

View-Only User Functions

5) Press the Search button.

Queries and Reports Help

Search for TPSs

TPS Query:

by TPS Number
 by Owner Username

Results

TPS Number	Cert Pack Rev	Cert Pack Iter	Cert Data Rev	Cert Data Iter	State	Owner Name
111-111-14	AA	5			Created	The admin
141-414-14			J	14	Created	John Doe

9. Glossary

JSP – JavaServer Pages, a technology which allows for the dynamic creation of web pages based on HTML using Java. ([EN.WIKIPEDIA.ORG/WIKI/JAVASERVER_PAGES](https://en.wikipedia.org/wiki/JavaServer_Pages))

EL – Expression Language, a tag-based language used in a JSP file which allows for the page it is on to dynamically access data from JavaBeans classes.

([DOCS.ORACLE.COM/JAVAAEE/6/TUTORIAL/DOC/BNAHQ.HTML](https://docs.oracle.com/javaee/6/tutorial/doc/bnahq.html))

JavaBeans – Classes used to encapsulate many objects into a single object which conforms to a particular convention. ([EN.WIKIPEDIA.ORG/WIKI/JAVABEANS](https://en.wikipedia.org/wiki/JavaBeans))

JDBC – An API to connect from Java code to a database.

([EN.WIKIPEDIA.ORG/WIKI/JAVA_DATABASE_CONNECTIVITY](https://en.wikipedia.org/wiki/Java_Database_Connectivity))

TPS – Test Program Sets, a set of instructions which go with the Spectrum CTS to allow the Spectrum CTS to test a specific airplane part.

Spectrum CTS – The Spectrum Common Test System is a functional test platform made by Teradyne to test commercial avionics.

10. References

Basham, B. Sierra, K. Bates, B. *Head First Servlets and JSP*. Tran. 2nd ed. O'Reilly, 2008. Print.

Chapple, Mike. "Database Normalization Basics." *About.com Databases*. 2012. Web.

<<http://databases.about.com/od/specificproducts/a/normalization.htm>>.

Christensen, Henrik. "Model-View-Controller Pattern." *Flexible, Reliable Software using Patterns and Agile Development*. Ed. Impagliazzo, John and McGettrick, Andrew. 6000 Broken Sound Parkway NW, Suite 300 Boca Raton, FL 33487: Chapman & Hall/CRC, 2010. 338-340. Print. Textbooks in Computing .

Eclipse.org. *Eclipse Indigo*. Tran. . Ed. . 3.7 Vol. The Eclipse Foundation, 2011. Print.

"JavaBeans." Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 29 August 2012.

Web. 30 Aug. 2012. <<http://en.wikipedia.org/wiki/JavaBeans>>

"Java EE Tutorial." Ed. . September 2010 2010. Web. August 2012

<<http://docs.oracle.com/javase/5/tutorial/doc/bnack.html>>.

"JDBC Documentation." Ed. . 2010. Web.

<<http://docs.oracle.com/javase/1.4.2/docs/guide/jdbc/>>.

Lubke, Ryan. Ball, Jennifer. Delisle, Pierre. "Unified Expression Language." *Sun Developer Network*. Ed. . August 2005 2005. Web.

<<http://java.sun.com/products/jsp/reference/techart/unifiedEL.html>>.

References

"SQL Query Reference." *SQL Tutorial*. Ed. . 2012.Web.

<<http://www.1keydata.com/sql/sql.html>>.

The Apache Software Foundation. *Tomcat*. Tran. . Ed. . 7.0.29 Vol. , 2011. Print.

Wheaton, Paul. "JavaRanch." *Coderanch*. Ed. . August 2012 2012.Web.

<<http://www.coderanch.com/forums/f-50/JSP>>.