TRAP

Travel Reimbursement Application Processing

Dylan Bettermann	Andrew Helgeson	Brian Maurer	Ethan Waytas
Bett0159@umn.edu	Helge206@umn.edu	Maure113@umn.edu	Wayt0012@umn.edu

10/15/2012
(*Updated: 11/7/2012*)

Requirements Document

Table of Contents

Introduction

Purpose

As a major research institution, the University of Minnesota has many professors, graduate students and others traveling domestically and internationally to attend conferences and remote locations for business. In order to support the advancement of knowledge, many different organizations and government programs provide grants that are awarded to professors, students, etc. These awardees can then claim expenses and be reimbursed for them. The expenses are entered into a form and then submitted to the accounting department for auditing and approval.

Due to the volume of expense submissions and the time required to audit each submitted form, the accounting department has requested a system that can autonomously receive and audit submitted reimbursement forms. This system would then send a form that has passed the auditing checks to the accountants for final approval. Such a system would save time and money, both for the accountants and the users, as errors in a form would be immediately sent to the user.

In actuality, the accounting department had contracted a system from a previous development team. This team created the front-end graphical user interface (GUI) that a user would interact with while inputting information into a reimbursement form. Due to unknown circumstances, the development team quit the project after developing the GUI (among some other things). What is now needed is the actual program to audit the forms after a user has submitted it.

The **T**ravel **R**eimbursement **A**pplication **P**rocessing (TRAP) program will need to take a reimbursement form from a user, audit it and determine if there are errors (and alert the user). If the form has been filled out correctly (e.g. date formats) with allowed expenses, it will then be sent to the accountants for final approval. This document will outline use cases and requirements that will help the users, accountants and developers to understand what the TRAP system needs to achieve with regards to increasing the accuracy, reliability and efficiency of auditing reimbursement forms.

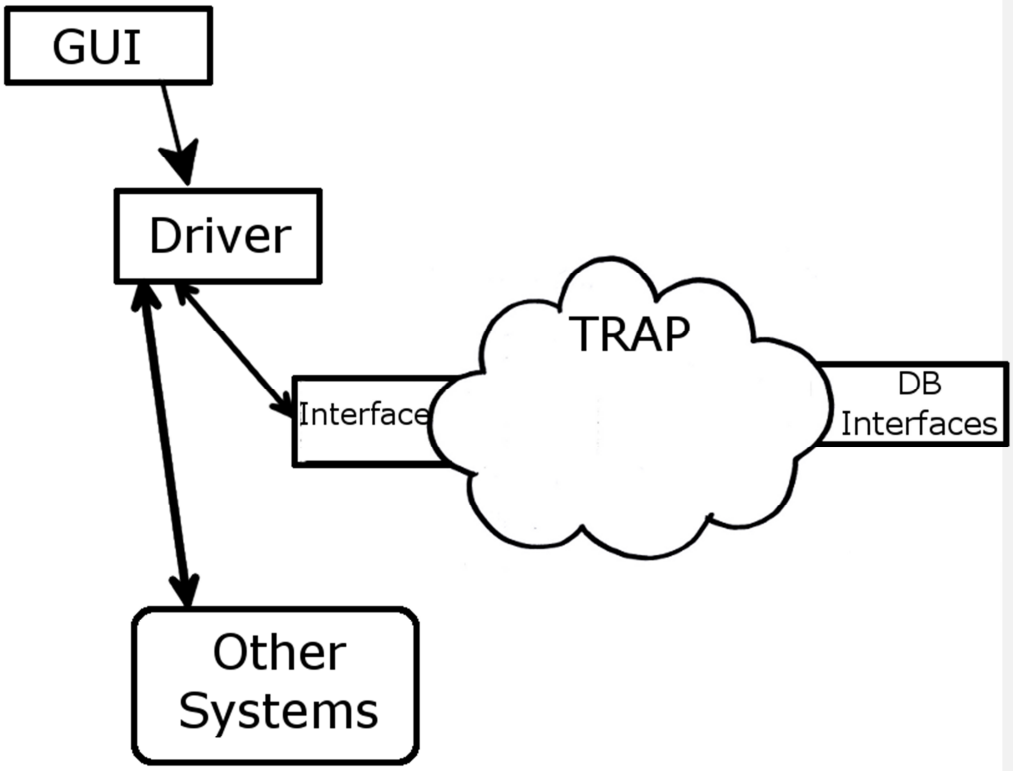
Intended Users

The TRAP system is intended for users who have been awarded grants and have eligible expenses that can be reimbursed through a grant. A sampling of potential users could be:

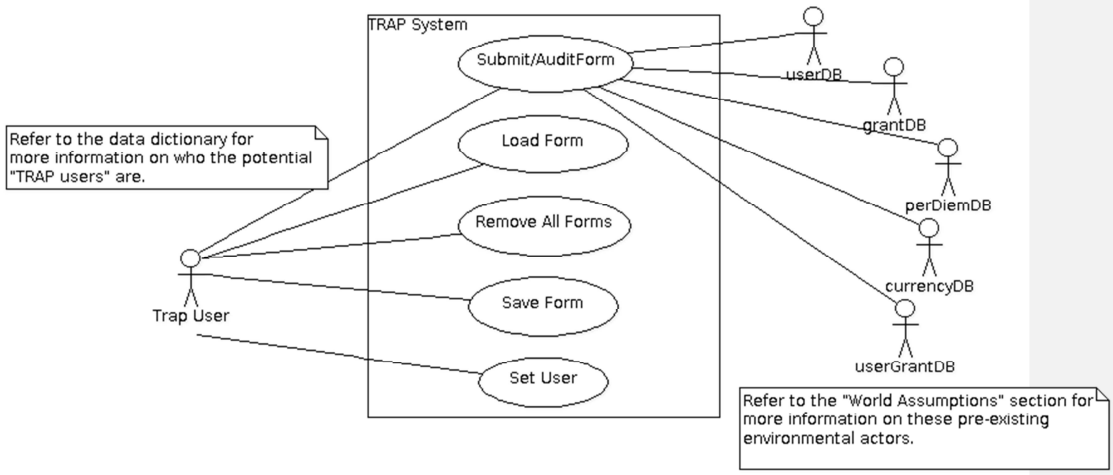
- A professor
- A graduate student
- A person awarded a grant from the University of Minnesota
    - This person may not necessarily be an employee of the University of Minnesota

In general, a user would be considered a person who has been awarded a grant and is able to claim expense reimbursement. Any potential claims for reimbursement will be viewed by an accountant who will have the final say on the release of money.

System Overview Diagram


GUI
Driver
TRAP
Interface
DB
Interfaces
Other
Systems

Use Case Diagram



Use Cases

Use Case #1

Use Case Name	**Save a Form**
Iteration	TBD
Summary	When requested through the *driver interface*, *TRAP* shall allow the saving of form data for the *current user* to be retrieved later.
Basic Course of Events	• *TRAP* receives a request through the *driver interface* to save form data under a provided *form id.* (see alternate paths below) • *TRAP* verifies that the provided *form id* is present under the set of saved forms for the *current user* and overwrites the data there with the provided form data. • Once completed, the provided form data is saved for later form loading (see use case #2) by the same user.
Alternative Paths	• If a form is submitted with a description it will be saved under a new *form id* by *TRAP*. This is the only way to generate a new *form id*.
Exception Paths	• If a *current user* has not been set with trap (UC #5), a *status error* shall be returned. • If a form is submitted with both a *form description* and a *form id* it will be rejected for saving. • If a form is submitted with just a *form id,* but that id is not present in *TRAP,* the form will be rejected for saving and a *status error* returned.
Extension Points	N/A
Trigger	The *TRAP* save form use case is initiated via the *driver interface* with form data and either a *form id* or *form description* as described above.
Assumptions	• The *TRAP user* has been properly authenticated and set as the *current user* in *TRAP*. • All appropriate data to be saved will be provided by the system invoking the save command. This input should correspond to the fields listed in the *TRAP Input Dictionary.*
Precondition	• The *TRAP* system is operational. • There is a *current user* set in *TRAP*.
Postcondition	The form data is saved under the requested *form id*. Whatever data was previously under that *form id* will be overwritten with the new save data. If the form data was saved with a description, the form's data will be saved under a new *form id* generated by *TRAP*. All saved forms shall be immediately available for loading (see use case #2) and shall persist for the duration that *TRAP* is running.
Author	Andrew Helgeson

Date	9/28/2012 (modified 10/13/2012)

Use Case #2

Use Case Name	**Load a saved form**
Iteration	TBD
Summary	When requested through the *driver interface*, *TRAP* shall be capable of retrieving the data of a saved form for the *current user*. For a form to be loadable it must have been previously saved (see use case #1).
Basic Course of Events	• A request is made to *TRAP* via the *driver interface* to load a form by its *form id*. The list of available forms for the given user can be retrieved from the *TRAP* system (see requirement 4.a - "Generate a list of loadable forms"). • *TRAP* will then retrieve and return all form data associated with the saved form. The returned data will be in the same format as described by the *TRAP Input Dictionary*.
Alternative Paths	As far as *TRAP* is concerned there is one path for loading a form (given *form id* to load).
Exception Paths	If a *current user* has not been set with trap (UC #5), a *status error* shall be returned. Since *TRAP* has no enforcement of external interfaces and systems it is possible an incorrect *form id* could be given to *TRAP*. In this case *TRAP* shall return a *status error*.
Extension Points	N/A
Trigger	The *form loading* use case is triggered by a load form call to the *TRAP* via the driver interface.
Assumptions	N/A
Precondition	• A *TRAP user* has successfully authenticated • The *TRAP user* has a set of saved forms that could potentially be loaded. If the *TRAP user* doesn't then the system falls into the exception path described above. • The *TRAP* system is operational
Postcondition	• The *TRAP* system is running • *TRAP* has successfully retrieved and returned the data associated with the saved form. • In the case of an invalid *form id, TRAP* will return a *status error*.
Author	Andrew Helgeson
Date	9/28/2012 (modified 10/13/2012)

Use Case #3

Use Case Name	**Remove all forms**
Iteration	TBD
Summary	Upon request for the driver, *TRAP* shall be able to remove all saved forms for the *current user*. A request through the driver interface to remove all *current user* forms shall be reflected in the *TRAP* system by a removal of all saved data and metadata the *current user*'s forms.
Basic Course of Events	• A request is made to *TRAP* through the *driver interface* to remove all form data and metadata for the *current user*. • *TRAP* locates all saved form data and metadata associated with *current user* and removes them. • *TRAP* sends a *status ok* to the service requestor.
Alternative Paths	As far as *TRAP* is concerned there is one path for removing a user's forms.
Exception Paths	If a *current user* has not been set with trap (UC #5), a *status error* shall be returned. In the case that an *invalid form id* is given to *TRAP* for removal, the request will be ignored and a *status error* returned.
Extension Points	*TRAP* could implement a secondary form removal interface which goes by *form id* for more selective removal. See use case #1 for more information on how forms are assigned ids.
Trigger	The *form removal* use case is triggered by a request to the *TRAP* driver interface for the removal of all forms associated with the *current user*.
Assumptions	• The *current user has been set*. If not, a *status error* will be returned as described in the exception paths.
Precondition	• A *TRAP user* has successfully authenticated and has been set as the *current user* within *TRAP*. • *TRAP* has received a request to remove all forms for the *current user*. • The *TRAP* system is operational.
Postcondition	• The *TRAP* system is operational. • If the *current user* is set then upon return all the user's saved form data and metadata will have been removed. After this action, if a request is made to get a list of loadable forms (see requirement 4a), no forms shall be returned unless a save form action has been initiated since. • If the *current user* is not set within *TRAP* a *status error* shall be returned.
Author	Andrew Helgeson
Date	9/28/2012 (modified 10/13/2012)

Use Case #4

Use Case Name	**Submit a *Reimbursement Form* to *TRAP* for *Auditing***
Iteration	TBD
Summary	A *TRAP user* shall be able to submit a *reimbursement form* for *auditing*.
Basic Course of Events	● The *TRAP system* receives a *form* from the *driver interface*. ● *TRAP* checks the *form* for existence (see Exception Paths below) ● *TRAP* will then begin to check input fields (see *TRAP Input Fields* dictionary) for correct formatting and validation ○ See requirements 1.a - 1.h ● *TRAP* will then check all *business logic* ○ See requirements 2.a - 2.n and 3.a - 3.f ● Once formatting, validation and *business logic* conditions have been checked and applied, the appropriate output shall be generated and returned via the *driver interface* (see *TRAP Output Fields* for a total list, along with requirements 5.c - 5.g). ○ This output will vary based on the various *business logic* rules that have been applied.
Alternative Paths	There is only one way for the *TRAP system* to receive a *form* for *auditing*.
Exception Paths	● If a *form* has been submitted without having been saved at least once, a *status error* shall be sent to the *driver interface*. ○ Else *auditing* shall continue ● If a *form* fails any of requirements 1.a - 1.h, a *status error* shall be returned. ● If a *form* fails any of the requirements in 2.a - 2.n and 3.a - 3.f, a *status error* shall be returned.
Extension Points	N/A
Trigger	The submit *form* for *auditing* is triggered by a *form* being sent to the *driver interface*.
Assumptions	None
Precondition	● A *TRAP user* has successfully authenticated ● The *TRAP* system is operational ● *TRAP* has received a *form* for *auditing*
Postcondition	● The *TRAP* system is operational ● *TRAP* has either successfully *audited* a *form* and returned and saved the output fields, or a *status error* has been returned upon failure of checking formatting/validation or *business logic*.
Author	Ethan Waytas
Date	9/28/2012 (modified 10/13/2012)

Use Case #5

Use Case Name	***Set User in TRAP***
Iteration	TBD
Summary	The TRAP system shall accept a user to be set for the context of later use cases.
Basic Course of Events	• The *TRAP system* receives a username as an x500 identifier via the *driver interface*. • The *TRAP* system verifies that the x500 is a valid user. • The *TRAP* system sets the current user to be the provided x500. • Later instances of use cases 1-4 will operate under the context of this user.
Alternative Paths	If there was a previously set *current user* and the requested value to set *current user* to is valid, the user being set will replace the *current user*.
Exception Paths	• If the provided x500 is not a valid user identifier or the user database is unavailable, *TRAP* shall return a *status error*. • While not an exception path of this use case, if this use case is not invoked and a valid user set before use cases 1-4 are operated a *status error* shall be generated.
Extension Points	N/A
Trigger	The driver (interface to TRAP) shall request that the *current user* of the TRAP system be set to the provided username.
Assumptions	We have access to a database with a list of users for checking. We do not assume however that this database will always be accessible.
Precondition	• The *TRAP* system is running
Postcondition	• The *TRAP* system is operational • In the case that the username to set is valid, *TRAP* shall set the current user to the provided username. If there was a previous current user there is no special action, they are simply replaced. • If the username is not valid or the database is down, *TRAP* shall leave the current user as is and return a *status error*.
Author	Andrew Helgeson
Date	11/1/2012

World Assumptions

- We assume that the active user of the *TRAP* system has been previously authenticated and the x500 username given to *TRAP* is valid for the current user.
- When the databases return a result it is assumed to be correct. This does not assume that the databases will always be available. We assume information has been accurately entered and the database system will serve it to us accordingly.
- The following databases are available to the *TRAP system*:
    - userDB
        - Provides the X500 number and information including the real name of the user, e-mail, citizenship, and visa status (if not a US citizen).
    - grantDB
        - Provides the account-specific information such as if it is sponsored/non-sponsored funding, the organization providing the sponsored funds, the type of organization (government, non-exportable, etc.), and the account balance.
    - perDiemDB
        - Provides the per diem/maximum daily rates for meals, incidentals, and lodging from the location.
    - currencyDB
        - Provides the currency exchange information for a given currency and date.
    - userGrantDB
        - The association between an account (in the grantDB) and users (in the userDB). It provides information on the account's admin, and authorized payees.
- We assume that *reimbursement forms* are being submitted for the reimbursement of expenses of a single traveler, regardless if all travelers are eligible under the listed grants. We must assume this because short of blacklisting more terms in justification fields, *TRAP* cannot reliably enforce this from the form input. With the other business policies placed on *reimbursement forms* it would make it very tough for an individual to claim the full expenses for multiple people in one form.
- We assume that the *TRAP* system will be called on through the *driver interface* in a non-concurrent manner. If *TRAP* is used in a concurrent fashion it cannot be guaranteed that correct results will be produced without proper synchronization. We have put this concurrent synchronization outside the scope of the development for this system.

Glossary of Terms

Average Developer: Assumed to be a developer who has spent 5 professional years on various software projects and has had some training and exposure to the *TRAP system*.

Business Logic: A set of business rules and policies which govern how the data of a *reimbursement form* is processed to generate a reimbursement amount. These policies also include what is not allowed and will cause a *reimbursement form* to be rejected.

Current User: When speaking about interactions with the *TRAP* system the *current user* is the user initiating some action. This user is assumed to have been authenticated outside of the *TRAP* system.

Database: A structured set of data in a computer that can be used for later retrieval.

Driver Interface: The interface the *TRAP* system uses to receive and send data. The input (receive) data is specified in the *TRAP Input Dictionary* and the output data is specified in the *TRAP Output Dictionary*.

Fatal Error: An error which prevents a *reimbursement form* from continuing through an *audit*.

Form Auditing (Audit, Auditing)**:** A *TRAP User* has submitted a *reimbursement form* to *TRAP* to determine if a user's reimbursement claims are acceptable. See use case case #4 and its related requirements for more information. Many of the *TRAP* system requirements are involved in the audit process.

Form Loading: The action where a *TRAP User* loads a saved, in-progress form from a list. Reference use case #2 and requirement 4.c for more information.

Form Removal: The action where a *TRAP User* selects and initiates the removal of an existing *reimbursement form* from the *TRAP* system. Reference use case #3 and requirement 4.f for more information.

Form Saving: The action where a *TRAP User* saves a *reimbursement form* within *TRAP* for later *form editing* and *form submission*. Reference use case #1 and requirement 4.e for more information.

Form Submission: The action where a *trap user* submits what they believe to be a completed form. This form is audited by the *TRAP* system and will return an error if the *reimbursement form* is incomplete or invalid. Reference use case #4 and requirement 4.d for more information.

Incidental Expense: The Federal Travel Regulation (www.gsa.gov/ftr) Chapter 300, Part 300-3, under Per Diem Allowance, describes incidental expenses as: A. Fees and tips given to porters, baggage carriers, bellhops, hotel maids, stewards or stewardesses and others on ships (per http://www.gsa.gov/portal/content/104208#12).

Justification: An explanation that shows a particular claim to be reasonable or necessary.

Module: A self contained entity of code that can be plugged in to the larger *TRAP system*.

Non-Fatal Error: An error that does not halt the auditing of a submitted *reimbursement form*.

Other Expenses: Expenses that are encountered on a trip that are not covered under incidentals, such as purchasing alcohol, buying an internet connection at a hotel, paying for meals for people other than the traveler, etc.

Reimbursement Form: The interface presented to the *TRAP User* requesting reimbursement. The input of this form is sent to the *TRAP* system for auditing. This form allows for *form loading, form saving, form editing,* and *form submission* through the *driver interface* to *TRAP.*

TRAP: The focus of this project which stands for ***T***ravel ***R***eimbursement ***A***pplication ***P***rocessing. *TRAP* is responsible for saving and loading in-progress *Reimbursement Forms* as well as auditing submitted forms.

TRAP User (User): A user of the *TRAP* system which could include any of the following people.

- University Faculty funded under a grant.
- Non-University faculty funded under a grant

Data Dictionary

Dictionary elements sorted alphabetically; additional elements or common names in parenthesis.

The data dictionary contains definitions of data elements reference throughout the requirements document. This list covers those elements which don't directly fall under the *TRAP Input Dictionary* or *TRAP Output Dictionary*. Combined, the input and output dictionaries as well as this dictionary define all data elements.

Database Timeout: The strict amount of time the *TRAP* system shall wait for a response from the *database*. See requirement 6.a for this timeout value and how to handle timeouts.

Form Description: A description for a given *reimbursement form.* See use cases 1 and 2 for more information on the description. See requirements 4.d and 4.f for uses of the *form description*.

Form Field: An area to enter information in the *reimbursement form*. More specifically to *TRAP* it refers to one of the potential input fields (see the *TRAP Input Dictionary)*.

Form ID: A unique identifier for a *reimbursement form* in the *TRAP* system. When the term *invalid form id* is used it means that either the *form id* does not exist in the *TRAP* system or it is not eligible for some operation. See use cases 1-4 and requirements 4.d - 4.f for uses of the *form id*.

TRAP Output: Upon successful *form auditing*, output data is generated and sent to the via the *driver interface*. See *TRAP Output Dictionary* for a list of output fields.

Status Error (error): A value returned from *TRAP* or used internally within *TRAP* that indicates some error. This error includes both a code for use by *TRAP* and a human readable message for display on the *Reimbursement Form*. See requirement 5.b for error handling information.

Status Ok: A value returned from *TRAP* or used internally within *TRAP* that defines an all good status.

TRAP Input Dictionary

These are high-level English descriptions of the data fields in a *reimbursement form*. Elements are sorted alphabetically except where they are dependent on another field (look for indentation).
Input field names with a *(n)* are tied to a specific day in the travel expense report or a numbered element in a list of expenses.

All input fields specified below at an indentation level of 0 (no indentation) are taken to be required fields for *TRAP*. Any input fields which are indented below one of these unindented fields is considered to be conditionally dependent upon the presence or value of its parent field. For example, ***Justification Presented*** is a required field and if its value is true then each of the sub-fields listed within it are also required.

Arrival Datetime: Date and timestamp of arrival at home area. See requirements 1.b - 1.d for format information.
Departure Datetime: Date and timestamp of departure from home area. See requirements 1.b - 1.d for format information.

Email Address: An email address where a *user* can be contacted.
Emergency Contact Name: Name of emergency contact.
Emergency Contact Phone: Phone number of emergency contact. Reference requirement 1.f for phone number formatting.

Justification for Travel: A justification for travel to the given conference.
Justification Conference Title: Event being traveled to.
Justification Presented: Is the traveler presenting?
- ***Justification Presentation Title:*** Title or presentation or paper.
- ***Justification Presentation Abstract:*** Abstract of presented work.
- ***Justification Presentation Acknowledgement:*** Grant acknowledgement in presentation/paper.

Justification Non-Sponsored: Justification for use of non-sponsored funds
Justification Sponsored: If using sponsored funding, a grant-specific statement of benefit

Num Days: Number of days of business travel.
- ***Breakfast City (n):*** City where breakfast was eaten on day *n*.
- ***Breakfast State (n):*** State where breakfast was eaten on day *n*.
- ***Breakfast Country (n):*** Country where breakfast was eaten on day *n*.
- ***Lunch City (n):*** City where lunch was eaten on day *n*.
- ***Lunch State (n):*** State where lunch was eaten on day *n*.
- ***Lunch Country (n):*** Country where lunch was eaten on day *n*.

Dinner City (n): City where dinner was eaten on day *n*.
Dinner State (n): State where dinner was eaten on day *n*.
Dinner Country (n): Country where dinner was eaten on day *n*.
Incidental City (n): City where incidentals were paid on day *n*.
Incidental State (n): State where incidentals were paid on day *n*.
Incidental Country (n): Country where incidentals were paid on day *n*.
Incidental Amount (n): Amount paid for incidentals on day *n*.
Incidental Currency (n): Currency paid for incidentals on day *n*.
Incidental Justification (n): Description of incidental expenses on day *n*.
Lodging City (n): City where traveler stayed on day *n*.
Lodging State (n): State where traveler stayed on day *n*.
Lodging Country (n): Country where traveler stayed on day *n*.
Lodging Amount (n): The amount spent on lodging on day *n*.
Lodging Currency (n): The currency paid for lodging on day *n*.

Num Grants: Number of accounts to be charged
Grant Account (n): Account number for funding source *n*.
Grant Percent (n): Percent of total cost to be charged to funding source *n*.

Num Other Expenses: Number of other expenses claimed
Other Expense Date (n): Date of expense *n.*
Other Expense Justification (n): Justification/Description of expense *n*.
Other Expense Amount (n): Amount paid for expense *n*.
Other Expense Currency (n): Currency paid for expense *n*.

Num Transportation: Number of transportation expenses claimed.
Transportation Date (n): Date of transportation expense *n*.
Transportation Type (n): The type of claimed transportation expense *n*. This includes types such as air, car, bus, train and so on.
Transportation Rental (n): Was transportation expense *n* a rental car?
Transportation Carrier (n): The name of the company contracted for transportation expense *n*.
Transportation Miles Traveled (n): If expense *n* is for gas mileage, how many miles were traveled.
Transportation Amount (n): Cost of transportation expense *n*.
Transportation Currency (n): Currency paid for transportation expense *n*.

Travel Type CSE Sponsored: Whether or not this travel is paid for with Computer Science & Engineering Department (CSE) funding.
Travel Type DTC Sponsored: Whether or not this travel is paid for with Digital Technology Center (DTC) sponsored funding.
Travel Type Non-Sponsored: Whether or not this travel is paid for with non-sponsored funds.

Username: X500 username

TRAP Output Dictionary

These are high-level English descriptions of the data fields in a *TRAP output*. Elements are sorted alphabetically.
Output field names with a *(n)* are tied to a specific day in the travel expense report or a numbered element in a list of expenses.
Output field names with a (d) may have multiple occurrences and are numbered by d

Arrival Datetime: Date and timestamp when the *user* arrived from the trip.
Citizenship: The current *user's* citizenship status.
Conference Title: Provides justification for conference attended by the *user.*

Number of Days: Total number of days on business travel.
- ***Day Date (n):*** Date on day *n.*
- ***Day Total (n):*** Total amount of money spent on day *n* in US dollars.
- ***Day Incidental Total (n):*** Total cost of incidentals on day *n* in US dollars.
- ***Day Incidental Justification (n):*** *User's* reasoning for an incidental on day *n* in US dollars.

Departure Datetime: Date and timestamp when the *user* left on the trip.

Number of Destinations: The number of destinations the user visited
- ***Destination City (n):*** Name of city visited on day *n*.
- ***Destination State (n):*** Name of state visited on day *n*.
- ***Destination Country (n):*** Name of country visited on day *n*.

Email Address: An email address where a *user* can be contacted.
Emergency Contact Name: Name of *user's* emergency contact.
Emergency Contact Number: Telephone number of the *emergency contact name*.
Form Submission Datetime: The *form's* submission timestamp.
Name: The user's full name.
Non-sponsored Justification: *User's* reason for applying a non-sponsored grant.

Number of Grants: Total number of grants to be charged.
- ***Grant Account Number (d):*** The account number of funding source *d*.
- ***Grant Percentage Charge (d):*** The percentage of the total to charge funding source *d*.
- ***Grant Amount to Charge (d):*** The amount to charge funding source *d* (as computed).
- ***Grant Approver Name (d):*** The name of the grant approver to approve charges for funding source *d*.

Number of Other Expenses: Total number of other expenses claimed.

Other Expense Date (n): Date of expense on day *n*.
Other Expense Justification (n): *User's* reasoning and description for expense on day *n*.
Other Expense Total (n): Total expenses on day *n* in US dollars.

Paid by University: Indicates whether or not a *user* is employed by the University of Minnesota.
Presentation: Is the *user* presenting at the conference?
Presentation Title: Title of the *presentation*.
Presentation Abstract: The abstract of the *user's presentation*.
Presentation Acknowledgment: A mention of the grant sponsor in the *presentation*.
Sponsored Justification: *User's* benefit from the grant.
Total Reimbursement: The total amount of expenses claimed.

Number of Transportation: Number of transportation expenses claimed.
Transportation Claims (total): Total number of expenses claimed for transportation.
Transportation Date (n): Date of transportation expense on day *n*.
Transportation Total Expense Claim (n): Total expense of transportation on day *n* in US dollars.
Transportation Type (n): Type of transportation on day *n*.

Username: X500 id of the *user*.
Visa Status: The current *user's* visa standing.

Individual Requirements

1. Form/Field Validation Requirements

Requirement 1.a

Title	**Forms shall only be sent for validation on submit (not on save).**
Number	**1.a**
Type	Primary and Essential
Use Case	Submit Form (#4)
Introduction	*TRAP* shall only validate *reimbursement forms* that a user has submitted.
Rationale	A *TRAP user* will generally save a *reimbursement form* for further editing or later retrieval. It has been determined that a *user* shall only be able to send a *reimbursement form* to the *TRAP system* for auditing upon clicking a "Submit" button (assumed to be present in the *TRAP GUI*). If a *reimbursement form* were to be automatically *submitted* on every save, many partially completed forms would be rejected by the *TRAP system*.
Author	Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	The user shall input information to the form, which will then be submitted to *TRAP* for *auditing*.
Requirement Description	A form shall be sent for validation by *TRAP* upon submission by the user. It is assumed that some mechanism for submission exists on the user's end (*TRAP GUI*).
Outputs	Firstly, it is assumed that a confirmation message is sent to the user by the *TRAP GUI* indicating that a certain *form* has been *submitted* for *auditing*. Output upon *form submission* shall be **one** of the following: ● A *form* that has successfully passed *auditing* shall be made available for approval by the accountants ● An *error* message indicating to the *user* where data has been entered incorrectly in the *form*
Persistent Change	--
User Satisfaction	5 (max)
User Dissatisfaction	5 (max)

Related Requirements	--
Conflicts	--
Support Materials	--
Test Cases	1.a

Requirement 1.b

Title	**Date format validation**
Number	**1.b**
Type	Primary and Essential
Use Case	Submit Form (#4)
Introduction	*TRAP* shall only accept dates that have been submitted with the correct format.
Rationale	*TRAP* must support a large number of users with many different backgrounds. In order to prevent confusion regarding dates (eg. dd-mm-yyyy, mm-dd-yyyy, etc.) , only one date format will be accepted.
Author	Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	The user shall input date information to the *reimbursement form*, which will then be submitted to *TRAP* for *auditing*.
Requirement Description	The date format shall be structured as follows: ● **yyyymmdd** ○ Years (yyyy) - 4 digits ○ Months (mm) - 2 digits (e.g. Jan = 01, Feb = 02 … Dec = 12) ○ Days (dd) - 2 digits
Outputs	Upon submission of a form containing a date: ● If the date is correctly formatted, it shall not be rejected and *auditing* will continue ● If the date is incorrectly formatted, an appropriate *error* message shall be returned to the user specifying the correct format
Persistent Change	--
User Satisfaction	5 (max)
User Dissatisfaction	5 (max)
Related Requirements	1.c, 1.d
Conflicts	--
Support Materials	--
Test Cases	1.b

Requirement 1.c

Title	**Datetime format validation**
Number	**1.c**
Type	Primary and Essential
Use Case	Submit Form (#4)
Introduction	*TRAP* shall only accept datetimes that have been submitted with the correct format.
Rationale	*TRAP* must support a large number of users with many different backgrounds. In order to prevent confusion regarding datetimes, only one datetime format will be accepted.
Author	Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	The *user* shall input datetime information to the *reimbursement form*, which will then be submitted to *TRAP* for *auditing*.
Requirement Description	The date format shall be structured as follows: ● **yyyymmdd<one space>hhmmss** ○ Years (yyyy) - 4 digits ○ Months (mm) - 2 digits (e.g. Jan = 01, Feb = 02 … Dec = 12) ○ Days (dd) - 2 digits ○ <one space> ■ Defined as one press of the spacebar ○ Hours (hh) - 2 digits (shall be in military time, 00 - 23) ○ Minutes (mm) - 2 digits (00 - 59) ○ Seconds (ss) - 2 digits (00 - 59)
Outputs	Upon submission of a *reimbursement form* containing a datetime: ● If the datetime is correctly formatted, it shall not be rejected and auditing will continue ● If the datetime is incorrectly formatted, an appropriate *error* message shall be returned to the *user* specifying the correct format
Persistent Change	--
User Satisfaction	5 (max)
User Dissatisfaction	5 (max)
Related Requirements	1.b, 1.d
Conflicts	--

Support Materials	--
Test Cases	1.c

Requirement 1.d

Title	**Date & Datetime range validation**
Number	**1.d**
Type	Primary and Essential
Use Case	Submit Form (#4)
Introduction	*TRAP* shall only accept dates and datetimes that are prior to *form submission* time. The range for dates and datetime must also be valid. There should be no arrival times before a corresponding departure time.
Rationale	The *TRAP user* shall be allowed to submit any date/time for the date or datetime as long it follows the specified format (Requirements 1.b and 1.c). From an accounting perspective, reimbursement cannot be claimed for travel which has not occurred yet. Logically, submitted date ranges such as travel departure and arrival times must be sane meaning there are no arrival dates or datetimes that come before their corresponding departure date or datetime.
Author	Andrew Helgeson, Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	The *user* shall input date or datetime information to the *reimbursement form*, which will then be *submitted* to *TRAP* for *auditing*.
Requirement Description	See Requirements 1.b and 1.c for strict date and datetime formatting. If these requirements are not met, *TRAP* will return a *status error* and halt *auditing*. Additionally, all date or datetimes must be prior to the current date a *reimbursement form* is *submitted*. Failure to meet this requirement shall cause an *error* to be sent to the *user*. *TRAP* shall also validate that the numbers for date months and days are valid. A date of 20120231 for example would be an invalid day for the month February. Similarly, 20121315 would be an invalid month for a date. If a provided date fails to meet this check *TRAP* shall return a *status error*. *TRAP* shall validate that the end datetime of a trip comes after the start datetime. If the provided datetimes do not follow this rule *TRAP* shall return a *status error*. *TRAP* shall also convert times specified as the n^{th} day of the trip to their appropriate datetime relative to the start datetime. This datetime for the n^{th} date must lie between the start datetime and the end datetime. It a provided day (n) falls outside the bounds of the start and end datetimes *TRAP* shall return a *status error*.
Outputs	Upon submission of a form containing a date or datetime: ● If the date or datetime is correctly formatted and prior to the date of *form*

	submission, it shall not be rejected and *auditing* will continue ● If the date or datetime is incorrectly formatted or a date in the future, an appropriate *error* message shall be returned to the *user* specifying the correct format ● If the n^{th} day is before <*arrival datetime*> or after <*departure datetime*> a *status error* shall be returned ○ Note, requirement 2.j is exempt from this
Persistent Change	--
User Satisfaction	5 (max)
User Dissatisfaction	5 (max)
Related Requirements	1.b, 1.c
Conflicts	--
Support Materials	--
Test Cases	1.d

Requirement 1.e

Title	**Required Form Fields**
Number	**1.e**
Type	Primary and Essential
Use Case	Submit Form (#4)
Introduction	*TRAP* shall only accept *reimbursement forms* with all the required *form fields* (and applicable sub-fields).
Rationale	In order for the *TRAP* system to correctly process a *form* there must be data for all *form fields* specified as requested.
Author	Andrew Helgeson, Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	A *form* to be *submitted* for *auditing* with the correctly formatted *form fields*.
Requirement Description	See *TRAP Input Dictionary* for a complete list of fields (and applicable sub-fields).
Outputs	If the *form fields* are correctly formatted and present, the *form* shall pass *auditing*. If *form fields* are not present or correctly formatted, a *status error* shall be returned to the *user*.
Persistent Change	--
User Satisfaction	5 (max)
User Dissatisfaction	5 (max)
Related Requirements	--
Conflicts	--
Support Materials	--
Test Cases	1.e

Requirement 1.f

Title	**Phone Number Fields**
Number	**1.f**
Type	Primary and Essential
Use Case	Submit Form (#4)
Introduction	Telephone numbers must be in a specific format.
Rationale	In order to have consistency, all telephone numbers must be inputted exactly the same.
Author	Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	A *form* for *auditing*, a telephone number.
Requirement Description	A telephone number shall be formated as the following: ● dddddddddd - d represents a digit between 0 and 9
Outputs	Upon submission of a form containing a telephone number: ● If the telephone number is correctly formatted, it shall not be rejected and *auditing* will continue ● If the telephone number is incorrectly formatted, an appropriate *error* message shall be returned to the *user* specifying the correct format
Persistent Change	--
User Satisfaction	5 (max)
User Dissatisfaction	5 (max)
Related Requirements	--
Conflicts	--
Support Materials	--
Test Cases	1.f

Requirement 1.g

Title	**Support for multiple currency formats**
Number	**1.g**
Type	Primary and Essential
Use Case	Submit Form (#4)
Introduction	*TRAP* shall be able to support different types of currency
Rationale	For *users* who travel outside the United States, expenses incurred might be paid in the local currency. To provide better functionality and reduce errors, *TRAP* will take currencies in multiple formats and do conversions during *auditing*.
Author	Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	A *form* for *submission* and with an expense amount and currency type.
Requirement Description	A currency type shall be specified as a string in the input field. Potential examples: ● EUR ● EURO ● POUND ● FRANC If a currency type is found to be empty, it shall be assumed to be the American dollar. Internally to *TRAP*, all currencies shall be converted to the dollar for calculations in grant deductions.
Outputs	Upon submission of a form containing a currency type: ● If the currency type is correctly formatted, it shall not be rejected and *auditing* will continue ● If the currency type is incorrectly formatted, an appropriate *status error* shall be returned to the *user* specifying the correct format
Persistent Change	--
User Satisfaction	5 (max)
User Dissatisfaction	5 (max)
Related Requirements	2.k
Conflicts	--

Support Materials	--
Test Cases	1.g

Requirement 1.h

Title	**Email address validation**
Number	**1.h**
Type	Primary and Essential
Use Case	Submit a *Reimbursement Form* to *TRAP* for *Auditing* (#5)
Introduction	*TRAP* shall only accept valid email addresses.
Rationale	A *user* may need to be contacted by an account for further explanation of expense claims. If an email is to be sent, the email address must be valid.
Author	Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	A *form* for *submission* and an <*email address*>.
Requirement Description	An <*email address*> shall be defined as follows: ● [local part]@[domain] ○ [local part] - typically a username ○ '@' - the "at" symbol standard in email addresses ○ [domain] - the place to send an email (e.g. umn.edu)
Outputs	Upon submission of a form containing an <*email address*>: ● If the <*email address*> is correctly formatted, it shall not be rejected and *auditing* will continue ● If the <*email address*> is incorrectly formatted or not present, an appropriate *error* message shall be returned to the *user* specifying the correct format
Persistent Change	--
User Satisfaction	5 (max)
User Dissatisfaction	5 (max)
Related Requirements	--
Conflicts	--
Support Materials	--
Test Cases	1.h

Requirement 1.i

Title	**At least 1 grant must be given and all grants must be valid**
Number	**1.i**
Type	Primary and Essential
Use Case	Submit a *Reimbursement Form* to *TRAP* for *Auditing* (#5)
Introduction	*TRAP* must have at least one valid grant provided in order to process the *reimbursement form*.
Rationale	Without at least one grant there is no point in processing the form because no reimbursements can be made.
Author	Andrew Helgeson
Source	TRAP Design Work, Wed Oct.. 31, 2012
Inputs	The grants provided with the form input. Access to the *grantDB* api.
Requirement Description	*TRAP* shall first check the validity of all provided grants in the *grantDB*. If there are any grants which are not found in the *grantDB* or otherwise return an error on lookup, *TRAP* shall return a *status error*. After *TRAP* has verified all grants it shall check that there is at least 1 grant. If there are no grants a *status error* shall be given. Otherwise a *status given* will be given.
Outputs	*Status Ok* - There are one or more grants and all provided grants are valid according to the *grantDB*. *Status Error* - There are no grants provided or there is at least one grant that is not valid according to the *grantDB*.
Persistent Change	--
User Satisfaction	5 (max)
User Dissatisfaction	5 (max)
Related Requirements	1.j
Conflicts	--
Support Materials	--
Test Cases	1.i

Requirement 1.j

Title	**Grant percentages must sum to no more than 100%**
Number	**1.j**
Type	Primary and Essential
Use Case	Submit a *Reimbursement Form* to *TRAP* for *Auditing* (#5)
Introduction	*TRAP* shall only accept a set of grants whose percentages sum to no more than 100%.
Rationale	*TRAP* is designed to calculate reimbursement to the amount of expenses that the applicant enters, no more and no less. Therefore it would be invalid to allow someone to specify a split in funding that exceeds the total of their reimbursement.
Author	Andrew Helgeson
Source	*TRAP* design work, Oct. 31, 2012
Inputs	A list of the grant percentage for each provided grant.
Requirement Description	*TRAP* shall sum the percentages provided for the set of grants. If the percentages sum to over 100% *TRAP* shall return a *status error*. If the percentages sum to 100% or less *TRAP* shall return a *status ok*.
Outputs	*Status Ok* - The sum of the percentages is equal to or less than 100%. *Status Error* - The sum of the percentages is greater than 100%.
Persistent Change	--
User Satisfaction	5 (max)
User Dissatisfaction	5 (max)
Related Requirements	1.i
Conflicts	--
Support Materials	--
Test Cases	1.j

Requirement 1.k

Title	***Transportation Miles Traveled* must be a positive integer**
Number	**1.k**
Type	Primary and Essential
Use Case	Submit a *Reimbursement Form* to *TRAP* for *Auditing* (#5)

Introduction	*TRAP* shall only accept <*Transportation Miles Traveled*> expense fields that are in integer format and greater than or equal to zero.
Rationale	Calculations for reimbursement of mileage traveled under a personal car are done on a fixed per-mile rate and partial miles are not counted. To prevent unexpected reimbursement amounts resulting from the truncation of a decimal number *TRAP* shall warn the user if the input is not an integer. The provided miles amount must also be greater than or equal to 0.
Author	Andrew Helgeson
Source	*TRAP* design work, Oct. 31, 2012
Inputs	All transportation expenses of the type *personal car*. Each *personal car* expense has a corresponding <*Transportation Miles Traveled*> field.
Requirement Description	*TRAP* shall check the that <*Transportation Miles Traveled*> field for all appropriate transportation expenses are in integer format If the mileage is not in integer format a *status error* shall be returned. Otherwise, if the mileage is an integer a *status ok* shall be returned.
Outputs	*Status Ok* - All <*Transportation Miles Traveled*> expense fields are in integer format. *Status Error* - One or more <*Transportation Miles Traveled*> expense fields are not an integer. One or more <*Transportation Miles Traveled*> expense fields are less than 0.
Persistent Change	--
User Satisfaction	5 (max)
User Dissatisfaction	5 (max)
Related Requirements	--
Conflicts	--
Support Materials	--
Test Cases	1.k

Requirement 1.l

Title	**Currency Abbreviation Format**
Number	**1.l**
Type	Primary and Essential
Use Case	Submit a *Reimbursement Form* to *TRAP* for *Auditing* (#5)
Introduction	*TRAP* shall only accept a currency field, i.e. <*Transportation Currency n*> in the proper

	format.
Rationale	To have a uniform format for currency types.
Author	Ethan Waytas
Source	*TRAP* design work, Oct. 31, 2012
Inputs	A currency field, i.e. <*Transportation Currency n*>
Requirement Description	A currency abbreviation shall be three (3) letters, i.e. (USD, EUR, GBP). If a currency abbreviation is left blank, it shall be assumed that it is for USD.
Outputs	*Status Ok* - The currency abbreviation is only three characters long or left blank *Status Error* - The currency abbreviation is not three characters long (excluding if left blank)
Persistent Change	--
User Satisfaction	5 (max)
User Dissatisfaction	5 (max)
Related Requirements	--
Conflicts	--
Support Materials	--
Test Cases	1.k

2. Travel Requirements

Requirement 2.a

Title	**Flights must be booked on U.S. air carriers**
Number	2.a
Type	Primary and Essential
Use Case	Submit a Reimbursement Form to TRAP for Auditing (#5)
Introduction	Flights claimed for travel reimbursement must be booked through a U.S. based carrier.
Rationale	This is required *business logic* that is enforced through federal law.
Author	Andrew Helgeson
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	• <*transportation type*> • <*transportation carrier*> • List of U.S. based carriers
Requirement Description	*TRAP* must check that all claimed transportation expenses with the <*transportation type*> corresponding to air travel have been booked with a U.S. based carrier. If any carriers are not U.S. based the form shall be rejected. It is assumed that the list of U.S. based flight carriers will be provided to the *TRAP* system.
Outputs	*Status ok* - If all claimed air carriers are U.S. based *Status error* - If any claimed air carriers are not U.S. based
Persistent Change	--
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	--
Conflicts	--
Support Materials	--
Test Cases	2.a

Requirement 2.b

Title	**Lodging reimbursements capped by per diem amount**
Number	2.b
Type	Primary and Essential
Use Case	Submit a Reimbursement Form to TRAP for Auditing (#5)
Introduction	*Trap users* submitting reimbursements for lodging are capped for reimbursement according to the per diem amount. Daily lodging expenses above the per diem amount will not be reimbursed.
Rationale	*Business logic* that must be implemented
Author	Andrew Helgeson
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	● <*Lodging amount (n)*> ● <*Lodging currency (n)*> ● <*Lodging Date*> ○ Since the other lodging information is tied to a particular day this will come with the <*Lodging amount (n)*> and <*Lodging currency (n)*> input. ● Per diem amount for lodging in USD (obtained from database)
Requirement Description	*TRAP users* requesting reimbursement are only allowed to collect a maximum per diem amount on travel lodging. Any amount above this per diem will not be reimbursed. It is assumed that the per diem amount taken as input is in USD currency. Before comparison to determine the reimbursement amount, the <*lodging amount*> must be converted from the <*lodging currency*> into USD at the exchange rate appropriate for the given <*lodging date*>. See requirement 2.k for proper currency conversion steps. The amount to reimburse is the minimum of the USD converted <*lodging amount*> and the per diem for lodging. This requirement only looks at a single instance of a lodging expense. For a *reimbursement form* with multiple lodging expenses this requirement must be applied for each claimed lodging expense.
Outputs	The amount of money (in USD) to be reimbursed for the given lodging expense claim. ● <*Day Total (n)*> shall be updated with the expense ○ In US dollars
Persistent Change	The calculated total for reimbursement should increase by the reimbursement amount outputted from this requirement logic.
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	2.k, 2.m, 2.n
Conflicts	**--**

Support Materials	**--**
Test Cases	2.b

Requirement 2.c

Title	**Only 75% of meal expenses reimbursed on first and last day**
Number	2.c
Type	Primary and Essential
Use Case	Form Submission
Introduction	On the first and last day(s) of travel only 75% of the per diem for meals can be reimbursed.
Rationale	This is a *business logic* policy that must be enforced.
Author	Andrew Helgeson
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	● The list of meals for the first and last travel day/s. ● The meal per diem from the *perDiemDB*
Requirement Description	Meals are normally reimbursed by the per diem rate. For the list of meals on the first and last day of the trip only 75% of this per diem will be reimbursed. To prevent *TRAP users* from claiming meals before their *departure time* on the day of travel we will enforce time bounds for when a certain meal can be claimed. See requirement 2.m for these bounds. It the reimbursement is for a one day trip 75% of the per diem will be reimbursed for that day.
Outputs	The amount to be reimbursed for the inputted meals. ● *<Day Total (n)>* shall be updated with the expense ○ In US dollars
Persistent Change	The inputted meals shall be earmarked in some grant and within *TRAP* they should be marked as processed. The running total for reimbursement within *TRAP* shall be updated to reflect this output.
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	2.m
Conflicts	--
Support Materials	**--**
Test Cases	2.c

Requirement 2.d

Title	**75% of incidentals on first and last day**
Number	2.d
Type	Primary and Essential
Use Case	Form Submission
Introduction	On the first and last day(s) of travel only 75% of the per diem for incidentals can be reimbursed.
Rationale	This is a *business logic* policy that must be enforced.
Author	Andrew Helgeson
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	● The list of incidentals for the first and last travel day/s. ● The incidentals per diem *perDiemDB*
Requirement Description	Incidentals are normally reimbursed by the per diem rate. For the set of incidentals on the first and last day/s of the travel, only 75% of this per diem will be reimbursed. It the reimbursement is for a one day trip 75% of the per diem will be reimbursed for that day.
Outputs	The amount to be reimbursed for the inputted incidentals.
Persistent Change	The inputted incidentals shall be earmarked in some grant and within *TRAP* they should be marked as processed. The running total for reimbursement within *TRAP* shall be updated to reflect this output.
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	2.n
Conflicts	--
Support Materials	**--**
Test Cases	2.d

Requirement 2.e

Title	**Only one (1) checked piece of luggage for air travel**
Number	2.e
Type	Primary and Essential
Use Case	Form Submission
Introduction	Only one checked back can be claimed for reimbursement for air travel.
Rationale	This is a *business logic* policy that must be enforced.
Author	Andrew Helgeson
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	● All claimed transportation expenses with the <*transportation type*> equivalent to baggage. The information required with this is: ○ <*Transportation Date*> ○ <*Transportation Amount*> ○ <*Transportation Currency*> ● All claimed transportation expenses with the <*transportation type*> equivalent to air travel. The information required with this is: ○ <*Transportation Date*> ● Start and end dates of the trip. See the requirement description below for more on this.
Requirement Description	Only one checked baggage can be claimed for reimbursement per day with the exception of single day trips in which case two will be allowed. The start and end dates of the trip are required to allow this exception. Each baggage expense shall be tallied according to the date of the expense. *TRAP* shall verify that there was an air travel expense on each day for a corresponding baggage expense on that day. If there are more baggage expenses than air travel expenses on a given day then the most expensive *n* baggage expenses will be considered for reimbursement. The fixed maximum for baggage expenses is $25. If <*transportation amount*> after currency conversion (see requirement 2.k) is greater than $25, *TRAP* shall only earmark $25 for reimbursement. If the claimed amount is less than $25 then that amount is used..
Outputs	The total amount to be reimbursed for checked baggage.
Persistent Change	The outputted reimbursement amount shall be earmarked in the appropriate grant/s. The reimbursement total within *TRAP* shall be incremented by the output amount.
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)

Related Requirements	2.j
Conflicts	--
Support Materials	**--**
Test Cases	2.e

Requirement 2.f

Title	**Family member expenses are not reimbursable**
Number	2.f
Type	Primary and Essential
Use Case	Form Submission
Introduction	Travelling expenses for the family of the submitting user are not reimbursable.
Rationale	This is a *business logic* policy that must be enforced.
Author	Andrew Helgeson
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	● The justification fields for all *other* and *incidental* expenses. ○ *<Other expense justification (n)>* ○ *<Incidental justification (n)>*
Requirement Description	Only the expenses of the *TRAP user,* who has been authenticated, are reimbursable. Since a majority of this is taken on the *TRAP user's* honesty, the *TRAP* system can only enforce this on justification fields for *other* and *incidental* expenses. *TRAP* shall enforce this policy through string searching of justification fields for terms referencing family. Included here is a non-inclusive list: wife, husband, child, son, daughter, father, mother
Outputs	*TRAP* shall return a *status ok* if no justifications contained family references according to our blacklist of terms. If any justifications contain a blacklisted term *TRAP* shall return a *status error*.
Persistent Change	This business logic policy solely performs a check.
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	2.g
Conflicts	--
Support Materials	**--**
Test Cases	2.f

Requirement 2.g

Title	**Car rental**
Number	2.g
Type	Primary and Essential
Use Case	Form Submission
Introduction	A *user* can claim expenses for a car rental.
Rationale	This is a *business logic* policy that must be enforced.
Author	Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	● <*Num transportation*> ● <*Transportation date*> ● <*Transportation type (n)*> ○ Shall be “rental car” ● <*Transportation rental (n)*> ○ Must be “True” ● <*Transportation carrier (n)*> ● <*Transportation amount (n)*> ● <*Transportation currency (n)*>
Requirement Description	If a *user* is claiming a car rental expense they must have filled out the inputs listed above. Car rental dates must be between the <*arrivale datetime*> and the <*departure datetime*>. If the date claimed does not fall within this range, a *status error* shall be returned. Additionally, a *user* cannot claim a rental car expense and expenses for a personal car on the same day.
Outputs	If any required inputs are not included, a *status error* shall be returned. If all data is available from the inputs, the following shall be set: ● <*Transportation claims (total)*> shall be increased by 1 ● <*Transportation date (n)*> ● <*Transportation type (n)*> type of transportation claimed on day *n* ● <*Transportation total expense claim (n)*>
Persistent Change	<*Transportation total expense claim (n)*> shall be updated with the expense claimed on day *n*.
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	2.h, 2.i

Conflicts	--
Support Materials	**--**
Test Cases	2.g

Requirement 2.h

Title	**Personal car**
Number	2.h
Type	Primary and Essential
Use Case	Form Submission
Introduction	A *user* may use a personal car for their trip and claim it as an expense.
Rationale	This is a *business logic* policy that must be enforced.
Author	Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	● <*Num transportation*> ● <*Transportation date*> ○ Must satisfy requirement 1.b ● <*Transportation type (n)*> ○ Shall be “personal car” ● <*Transportation rental (n)*> ○ Must be “No” ● <*Transportation miles travelled (n)*>
Requirement Description	If a *user* is claiming a personal car expense they must have filled out the inputs listed above. Personal car dates must be between the <*arrival datetime*> and the <*departure datetime*>. If the date claimed does not fall within this range, a *status error* shall be returned. Additionally, a *user* cannot claim a rental car expense and expenses for a personal car on the same day.
Outputs	If any required inputs are not included, a *status error* shall be returned. If all data is available from the inputs, the following shall be set: ● <*Transportation claims (total)*> shall be increased by 1 ● <*Transportation date (n)*> ● <*Transportation Total Expense Claim (n)*> total amount of rental expenses claimed on day *n* provided from the input ○ For <*Transportation miles traveled (n)*>, 1 mile shall equal $0.55 toward the <*Transportation total expense claim (n)*> ● <*Transportation type (n)*> type of transportation claimed on day *n*
Persistent Change	<*Transportation total expense claim (n)*> shall be updated with the expense claimed on day *n*.
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)

Related Requirements	2.g, 2.i
Conflicts	--
Support Materials	**--**
Test Cases	2.h

Requirement 2.i

Title	**Supported transportation types**
Number	2.i
Type	Primary and Essential
Use Case	Form Submission
Introduction	*TRAP* shall be able to support a variety of approved transportation types.
Rationale	This is a *business logic* policy that must be enforced.
Author	Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	• <*Transportation type (n)*>
Requirement Description	The type of transportation on day *n*. The only accepted types are as follows: • air • personal car • rental car • rail • taxi • public transportation • gas • parking • luggage
Outputs	If <*Transportation type*> is one of the accepted above types, *auditing* shall continue. Otherwise a *status error* shall be returned.
Persistent Change	--
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	--
Conflicts	--
Support Materials	**--**
Test Cases	2.i

Requirement 2.j

Title	**'Other' expenses**
Number	2.j
Type	Primary and Essential
Use Case	Form Submission
Introduction	A *user* may incur expenses unrelated to a trip, such as purchasing alcohol, internet, paying for meals for someone else, etc.
Rationale	This is a *business logic* policy that must be enforced.
Author	Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	● *<Num other expenses>* ● *<Other date (n)>* ● *<Other justification (n)>* ● *<Other amount (n)>* ● *<Other currency (n)>*
Requirement Description	A *user* claiming other expenses must provide a justification for later approval. See requirement 3.e and 3.f. If an *other expense* is not allowed, a *status error* shall be returned. The above inputs must be present, or a *status error* shall be returned. If an *other expense* is a meal for a person other than the *user*, alcohol, or internet, it shall have a claim date between *<departure datetime>* and *<arrival datetime>*. Expenses not listed above must have a claim date before the date a *user submits* a *form*. This means that a *user* is allowed to claim an *other expense* before the *<departure datetime>*.
Outputs	● *<Number of other expenses>* ● *<Other expense date (n)>* ● *<Other expense justification (n)>* ○ On day *n* ● *<Other expense total (n)>* ○ On day *n*
Persistent Change	*<Other expense total (n)>* will be updated with the new total for 'other' expenses on day n.
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	--

Conflicts	--
Support Materials	**--**
Test Cases	2.j

Requirement 2.k

Title	**Proper Currency Conversion**
Number	2.k
Type	Primary and Essential
Use Case	Form Submission
Introduction	*TRAP* needs to be able to support multiple currency formats.
Rationale	This is a *business logic* policy that must be enforced.
Author	Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	Any input field that has an expense amount (e.g. <*Lodging amount (n)*>) shall also have an associated currency field (e.g. <*Lodging currency (n)*>) and date field (e.g. <*Lodging date (n)*>).
Requirement Description	A currency abbreviation will be provided by the *user* in the appropriate currency field. This abbreviation shall be looked up in the currencyDB (see World Assumptions) for the appropriate conversion rate for the date the expense occurred. For any expense claimed, there is an associated date field (eg requirement 2.h). The *currencyDB* shall provide currency conversion rates for multiple dates. The expense amount claimed shall be converted to US dollars using the appropriate conversion rate for the date the expense occurred. If a currency field is left blank, it shall be assumed the currency is in US dollars.
Outputs	If a query to the currencyDB does not return a result, a *status error* shall be returned. If a query to the currencyDB does return a result (this result is assumed to be correct, see World Assumptions) the amount field will be converted using the conversion rate for the date of the expense.
Persistent Change	--
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	6.a
Conflicts	--
Support Materials	**--**

Test Cases	2.k

Requirement 2.l

Title	**Transportation Mileage**
Number	2.l
Type	Primary and Essential
Use Case	Form Submission
Introduction	A *user* can claim expenses for mileage while using a personal car.
Rationale	This is a *business logic* policy that must be enforced.
Author	Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	● <*Transportation type (n)*> ○ Must be “personal car” ● <*Transportation rental (n)*> ○ Must be “false” ● <*Transportation miles traveled (n)*>
Requirement Description	A *user* will submit the total number of miles traveled in their personal car on day *n*. This number shall be a whole number. Expense claim will be calculated as: $0.55 per mile
Outputs	If the <*Transportation mile traveled (n)*> field is not a whole number or is not submitted with a personal car claim, a *status error* shall be returned. If <*Transportation miles traveled (n)*> is correctly formatted, the expense shall be calculated at $0.55 per mile. Output field <*Transportation total expense claim (n)*> shall be updated.
Persistent Change	The <*Transportation total expense claim (n)*> output field shall be updated according to the computation described in this requirement.
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	2.h
Conflicts	--
Support Materials	**--**
Test Cases	2.l

Requirement 2.m

Title	**Meal per Diem**
Number	2.m
Type	Primary and Essential
Use Case	Form Submission
Introduction	A *user* is entitled to a meal per diem.
Rationale	This is a *business logic* policy that must be enforced.
Author	Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	We need the information for one meal which can consist of one of the following set of inputs: ● *<Breakfast City (n)>* ● *<Breakfast State (n)>* ● *<Breakfast Country (n)>* *or* ● *<Lunch City (n)>* ● *<Lunch State (n)>* ● *<Lunch Country (n)>* *or* ● *<Dinner City (n)>* ● *<Dinner State (n)>* ● *<Dinner Country (n)>* In the end, this *business logic* will be applied multiple times for each claimed meal.
Requirement Description	In order for a *user* to claim their per diem, the *user* must eat at least one meal per day of their trip. The date of a meal shall be calculated from the departure date using the inputted day *n*. This date must fall between the *<departure datetime>* and the *<arrival datetime>*. The following times shall be used to determine if a *user* is claiming a per diem before leaving on a trip or after returning (all times are local and shall be converted to the time of the departure location for the *user):* ● Breakfast ○ 4:00 am - 11:00 am ● Lunch ○ 11:00 am - 4:00 pm ● Dinner ○ 4:00 pm - 4:00 am The meal per diem shall be determined from the *perDiemDB*.
Outputs	If the meal claimed is claimed before *<departure datetime>* or after *<arrival datetime>*, a *status error* shall be returned. ● If a meal is claimed on the day of departure or arrival, that meal will be

	compared to the meal times listed above. ○ If the meal is before <*departure datetime*>, a *status error* shall be returned. ○ If the meal is after <*arrival datetime*>, a *status error* shall be returned. Otherwise <*Day Total (n)*> shall be updated with the per diem expense in USD.
Persistent Change	If *TRAP* did not result in a *status error* while processing the meal claim, <*Day Total (n)*> shall be updated with the addition of the meal per diem reimbursement.
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	--
Conflicts	--
Support Materials	**--**
Test Cases	2.m

Requirement 2.n

Title	**Incidental per Diem**
Number	2.n
Type	Primary and Essential
Use Case	Form Submission
Introduction	A *user* may experience various *incidental expenses* and is entitled to reimbursement.
Rationale	This is a *business logic* policy that must be enforced.
Author	Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	● *<Incidental City (n)>* ● *<Incidental State (n)>* ● *<Incidental Country (n)>* ● *<Incidental Amount (n)>* ● *<Incidental Currency (n)>* ● *<Incidental Justification (n)>*
Requirement Description	A *user* shall provide an *incidental expense* with the above inputs. If any of the inputs are blank or omitted, a *status error* shall be returned. The incidental amount and currency shall be calculated according to requirement 2.k. If this requirement is not met, a *status error* shall be returned. The incidental per diem shall be determined from the perDiemDB.
Outputs	● *<Day date (n)>* ○ To be set as day *n* from the *<departure datetime>* ● *<Day Incidental Total (n)>* ○ Updated with the cost of the incidental on day *n* ● *<Day Incidental Justification (n)>*
Persistent Change	The *<Day incidental Total (n)>* should be incremented by the calculated per diem for the incidental expense.
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	--
Conflicts	--
Support Materials	**--**

Test Cases	2.n

3. Grant Requirements

Requirement 3.a

Title	**User splitting reimbursement across multiple grants**
Number	3.a
Type	Primary and Essential
Use Case	Submit a Reimbursement Form to TRAP for Auditing (#4)
Introduction	A *TRAP user* shall be able to split reimbursement funding across multiple grants.
Rationale	A *TRAP user* may be eligible for reimbursement under several grants and for any number of reasons they may want to split their reimbursement across several grants.
Author	Andrew Helgeson
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	● List of *grant accounts* ● List of *grant percentages* for each respective account ● A *grantDB* to retrieve grant restrictions necessary for partitioning.
Requirement Description	It is assumed that any form data being provided to this requirement has already passed form validation and verification. See requirement section 1 for more information. After *TRAP* has completed its audit of the *reimbursement form* it shall have a total reimbursement amount. The total reimbursement amount shall be partitioned according to the *grant percentages.* *TRAP* must be aware of grant specific expenses during partitioning. For example, requirement 3.e specifies that alcohol is only allowed under non-sponsored grants. If the *TRAP user* requests a split across sponsored and non-sponsored grants it must be ensured that the expenses for alcohol get earmarked under the non-sponsored grant. Expenses which are not grant specific can be carried over across multiple grants during partitioning. This requirement does not involve the earmarking of funds. See requirement 3.b.
Outputs	*TRAP* shall output a list of funds required from each grant according to the grant percentages to fulfill the total reimbursement amount calculated by *TRAP*.
Persistent Change	N/A
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	3.b (The actual earmarking of these funds)
Conflicts	--
Support Materials	--
Test Cases	3.a

Requirement 3.b

Title	**"Hold" funds after successful audit**
Number	3.b
Type	Primary and Essential
Use Case	Submit a Reimbursement Form to TRAP for Auditing (#4)
Introduction	*TRAP* shall mark funds as held before successfully completing a form audit.
Rationale	This is required so that funds are guaranteed to be available come later approval.
Author	Andrew Helgeson
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	• List of *grant accounts* • A *grantDB* to find available grant funds • The grant funding partition generated from requirement 3.a.
Requirement Description	Before accepting the processing of a *reimbursement form*, *TRAP* must earmark the funds required for the reimbursement.. Given the partition of funds by grant generated from requirement 3.a *TRAP* shall proceed to check and earmark the funds in each respective grant. If a grant is found without the appropriate funds as defined by the partition, all previously held funds for this reimbursement shall be released and *TRAP* shall return a *status error*. Released funds means that these funds are added back to the available money in the *grantDB* to be used for other transactions. If all grants have the appropriate funding the funds shall be earmarked and *TRAP* shall return a *status ok*.
Outputs	*Status ok* - If all grants have the required funds specified in the funding partition. *Status error* - If any grant does not have the required funds according to the funding partition.
Persistent Change	In the case of a *status ok* result, the *grant accounts* specified for funding shall have the funds specified in the funding partition for this reimbursement earmarked. In the case of a *status error* result, the *grant accounts* specified for funding shall not have any earmarked funds as a result of this reimbursement.
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	3.a
Conflicts	--
Support Materials	--
Test Cases	3.b

Comment [1]: Andrew Helgeson:
review

wayt0012:
reviewed

Comment [2]: wayt0012:
What about the case where a user does not provide a percentage, but has multiple grants available? Would TRAP then automatically try to figure out how to deduct from the grants, or would we return a status error?

wayt0012:
Say we had an example where a user could not charge their grants as they indicated with the percentages. Would TRAP be required to send a status error, or would TRAP try to figure out how to make things work with different grants. Right now you have the status error situation, which I believe is the correct response as the user is probably keeping track of how much money they have left in their grants (and would be surprised to find their records differ from the actual amounts), but it is something to think about.

Requirement 3.c

Title	**Grant approver name**
Number	3.c
Type	Primary and Essential
Use Case	Submit a Reimbursement Form to TRAP for Auditing (#4)
Introduction	*TRAP* shall provide the grant approver names with the output from the form audit.
Rationale	The name of the grant's funding manager is necessary for accounting to get approval to release funds for reimbursement later. The automated addition of these names will save considerable time for personnel later.
Author	Andrew Helgeson
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	• A list of *grant accounts*. Only those which resulted with earmarked funds are actually needed. • A *grantDB* to obtain grant approver names
Requirement Description	*TRAP* should provide, along with the regular audit output (see requirement 5.c), a list of grant approver names for all grants which had funds held as a result of the *reimbursement form* processing. *TRAP* will obtain these names from the *grantDB*. If the name of the grant approver is the same as the user submitting the form *TRAP* shall not provide a grant approver name for this grant.
Outputs	The list of grant funding approver names generated according to the rules in the requirements description.
Persistent Change	--
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	3.b
Conflicts	--
Support Materials	--
Test Cases	3.c

Requirement 3.d

Title	**Alcohol allowed under non-sponsored grant**
Number	3.d
Type	Primary and Essential
Use Case	Form Submission (#4)
Introduction	Alcohol can only be claimed for reimbursement under a non-sponsored grant.
Rationale	This is a *business logic* policy that must be enforced.
Author	Andrew Helgeson
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	● *<Other Expense Date>* ● *<Other Expense Justification>* ● *<Other Expense Amount>* ● *<Other Expense Currency>* ● *<Grant Account>* ○ We need the grant account for each grant listed on the *reimbursement form* ○ Assumed with this is a database of grants where more information can be obtained for this account.
Requirement Description	All sponsored grants restrict the reimbursement of alcohol. Non-sponsored grants are free of this restriction and will permit the reimbursement of alcohol expenses. It is assumed from the honesty of the *TRAP user* that alcohol will be listed under an *other expenses* category which has a *justification* field. The only method for enforcement of this rule is by basic string matching of the justification field for the word 'alcohol' or obviously related terms such as 'rum', 'whiskey' etc. Since this is a loose enforcement it is up to the implementer to decide the list of terms to search to use. If the <*other expense justification*> does not involve alcohol according to the procedure described above then return a *status error* indicating that this business logic requirement does not apply to the given expense. This is not a fatal, form rejecting error. If an <*other expense justification*> is found to have references to alcohol the first step is to check the available grant types to find a non-sponsored grant. If none is found then reject the expense and the form. If one or more non-sponsored grants is found continue to below. First, the <*other expense amount*> must be converted to USD according to the exchange rate of the <*other expense currency*> at the time <*other expense date*>. If a non-sponsored grant is found to have the available funds required for this expense then earmark it. If none is found with sufficient funds then reject the expense and the form.
Outputs	**Status Ok** - If the justification was found to refer to alcohol and there were sufficient funds in a listed non-sponsored grant to cover the currency converted amount. ***Status Error***: ● If the expense specified is not found to relate to alcohol. This should be a *non-fatal error* and only means to look elsewhere to process it. ● If the expense is found to involve alcohol and either no non-sponsored grants were found or none with funding, an error should be returned that will reject the expense and the form.

Persistent Change	The following persistent state changes only apply when the requirement outputs a *status ok* as specified above. ● The non-sponsored grant selected to fund the expense should have the calculated amount as specified above earmarked. ● This expense should be marked as completed and not processed again.
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	--
Conflicts	--
Support Materials	--
Test Cases	**3.d**

Requirement 3.e

Title	**Internet Only Under Non-Sponsored Grants**
Number	3.e
Type	Primary and Essential
Use Case	Submit a Reimbursement Form to TRAP for Auditing (#4)
Introduction	Internet access costs while traveling are not reimbursable under sponsored grants.
Rationale	This is a *business logic* policy that must be enforced.
Author	Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	• *<Other Expense Date>* • *<Other Expense Justification>* • *<Other Expense Amount>* • *<Other Expense Currency>* • *<Grant Account>* o We need the grant account for each grant listed on the *reimbursement form* o Assumed with this is a database of grants where more information can be obtained for this account.
Requirement Description	All sponsored grants restrict the reimbursement of internet access purchases. Non-sponsored grants are free of this restriction and will permit the reimbursement of internet related expenses. It is assumed from the honesty of the *TRAP user* that internet access will be listed under an *other expenses* category which has a *justification* field. The only method for enforcement of this rule is by basic string matching of the justification field for the word 'internet'. Since this is a loose enforcement it is up to the implementer to decide the list of terms to search to use. If the *<other expense justification>* does not involve internet access according to the procedure described above then return a *status error* indicating that this business logic requirement does not apply to the given expense. This is not a fatal, form rejecting error. If an *<other expense justification>* is found to have references to internet access the first step is to check the available grant types to find a non-sponsored grant. If none is found then reject the expense and the form. If one or more non-sponsored grants is found continue to below. First, the *<other expense amount>* must be converted to USD according to the exchange rate of the *<other expense currency>* at the time *<other expense date>*. If a non-sponsored grant is found to have the available funds required for this expense then earmark it. If none is found with sufficient funds then reject the expense and the form.
Outputs	**Status Ok** - If the justification was found to refer to internet acces and there were sufficient funds in a listed non-sponsored grant to cover the currency converted amount. ***Status Error**:* • If the expense specified is not found to relate to internet access. This should be a *non-fatal error* and only means to look elsewhere to process it. • If the expense is found to involve internet access and either no non-sponsored grants

	were found or none with funding, an error should be returned that will reject the expense and the form.
Persistent Change	--
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	--
Conflicts	--
Support Materials	--
Test Cases	3.e

Requirement 3.f

Title	**No-export Grants only for US Citizens**
Number	3.f
Type	Primary and Essential
Use Case	Submit a Reimbursement Form to TRAP for Auditing (#4)
Introduction	US citizenship is required for grants that specify “no-export”.
Rationale	This is a *business logic* policy that must be enforced.
Author	Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	• List of *grant accounts* • A *grantDB* to find available grant types • A *userDB* to determine the citizenship of the *current user*.
Requirement Description	The *current user* of the *TRAP system* must have US citizenship to use grants with the type “noExport”. The grant type shall be determined from the *grantDB* and the citizenship of the *current user* shall be determined from the *userDB*. If the *current user* is a US citizen and the grant type is “noExport”, the *current user* is eligible for reimbursement under that grant.

	Else the *current user* is not eligible for reimbursement under that grant.
Outputs	***Status OK*** - If the *current user* is able to use a "noExport" grant ***Status Error****:* • If the *current user* is unable to be reimbursed under a "noExport" grant and has other grants they are charging.. This should be a *non-fatal error* and only means to look elsewhere to process it. • If the *current user* is unable to be reimbursed under a "noExport" grant and has no other grants available to charge, an error shall be returned that will alert the user to a lack of grants.
Persistent Change	--
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	3.b
Conflicts	--
Support Materials	--
Test Cases	3.f

Requirement 3.g

Title	**Foreign Grants, No Domestic Travel**
Number	3.g
Type	Primary and Essential
Use Case	Submit a Reimbursement Form to TRAP for Auditing (#4)
Introduction	Foreign grants do not pay for domestic travel (in the US).
Rationale	This is a *business logic* policy that must be enforced.
Author	Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	• A list of *grant accounts*. • A *grantDB* to obtain grant types.

	• A list of transportation expenses.
Requirement Description	If a grant type is “foreign”, domestic travel (within the US) shall not be reimbursable. Requirements 2.a, 2.e, 2.g, 2.h, 2.i and 2.l become ineligible.
Outputs	If any expenses related to requirements 2.a, 2.e, 2.g, 2.h, 2.i or 2.l are charged to a grant with grant type of “foreign” a *Status Error* shall be returned with the following: • If there are other grants available to charge to, this error shall be *non-fatal* and *auditing* shall continue. • If there are no other grants available to charge, this error shall be returned to the user along with an appropriate *error message*.
Persistent Change	--
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	3.b
Conflicts	--
Support Materials	--
Test Cases	3.g

Requirement 3.h

Title	**NIH Grant Restrictions**
Number	3.h
Type	Primary and Essential
Use Case	Submit a Reimbursement Form to TRAP for Auditing (#4)
Introduction	*TRAP* shall enforce business logic restrictions specific to the use of NIH grants.
Rationale	Grants provided by the NIH have specific restrictions that must be enforced.
Author	Andrew Helgeson
Source	Group Meeting, Friday Sept. 28, 2012 (Written Nov. 5, 2012)
Inputs	• A list of *grant accounts*.

	• A list of all expenses for the current *reimbursement application*.
Requirement Description	When an NIH grant is present there are a set of restrictions which *TRAP* shall enforce: • NIH grants shall not reimburse for food (meal) expenses. • NIH grants shall not reimburse for any transportation expense except air travel and public transit.
Outputs	If there is an expense for which an NIH grant cannot reimburse and there are no other available source grants, a *status error* shall be returned. Otherwise, any expenses for which the NIH grant cannot reimburse shall be marked as reimbursable only under the grants that allow it.
Persistent Change	--
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	--
Conflicts	--
Support Materials	--
Test Cases	3.h

Requirement 3.i

Title	**DoD Grant Restrictions**
Number	3.i
Type	Primary and Essential
Use Case	Submit a Reimbursement Form to TRAP for Auditing (#4)
Introduction	*TRAP* shall enforce business logic restrictions specific to the use of DoD grants.
Rationale	Grants provided by the DoD have specific restrictions that must be enforced.
Author	Andrew Helgeson
Source	Group Meeting, Friday Sept. 28, 2012 (written Nov. 5, 2012)
Inputs	

	• A list of *grant accounts*. • A list of all expenses for the current *reimbursement application*.
Requirement Description	When an DoD grant is present there are a set of restrictions which *TRAP* shall enforce: • DoD grants shall not reimburse for breakfast meal expenses. • DoD grants shall only reimburse for rental car expenses through the Hertz rental car provider. • DoD grants shall not reimburse for any non-domestic travel expenses.
Outputs	If there is an expense for which an DoD grant cannot reimburse and there are no other available source grants, a *status error* shall be returned. Otherwise, any expenses for which the DoD grant cannot reimburse shall be marked as reimbursable only under the grants that allow it.
Persistent Change	--
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	--
Conflicts	--
Support Materials	--
Test Cases	3.i

Requirement 3.j

Title	**Restrictions on Domestic Car Rental**
Number	3.j
Type	Primary and Essential
Use Case	Submit a Reimbursement Form to TRAP for Auditing (#4)
Introduction	Only one domestic car rental agency is allowed for reimbursement (National Traveler) unless superseded by requirement 3.i.
Rationale	This is a *business logic* policy that must be enforced.
Author	Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012

Inputs	• A list of *grant accounts*. • A list of all expenses for the current *reimbursement application*. • A car rental expense claim as specified in requirement 2.g. o Specifically, <*Transportation carrier (n)*>
Requirement Description	Unless the grant type is “DoD” (see requirement 3.i), the <*Transportation Carrier (n)*> field shall have “National Traveler” as the carrier for domestic car rental (US).
Outputs	If the carrier is “National Traveler” and the grant type being charged is not the “DoD”, then the expense is eligible for reimbursement (must also meet requirement 2.g). The list of eligible grants shall be added to a restricted line item. If the carrier is not “National Traveler” an empty restricted line item shall be created. If the carrier is “National Traveler” and the grant type is “DoD”, see requirement 3.i.
Persistent Change	--
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	3.b
Conflicts	--
Support Materials	--
Test Cases	3.j

4. Form State Requirements

Requirement 4.a

Title	**Generate a list of loadable forms**
Number	**4.a**
Type	Primary and Essential
Use Case	Load a Saved Form (#4)
Introduction	*TRAP* should be able to generate a list of all possible *reimbursement forms* that a given *user* has access to.
Rationale	The *TRAP GUI* needs to be able to display this list, so a request shall be made from it to *TRAP*.
Author	Brian Maurer, Andrew Helgeson, Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	The *user* requesting the list
Requirement Description	The list of loadable forms shall be generated and returned to the *TRAP GUI* if and only if: ● The *user* is logged in ● The specified *user* has loadable *forms* ○ A loadable *form* shall have a username associated with it. ○ For a *user* to load a *form*, the username provided by the *user* must match the username associated with the *form*.
Outputs	If the requirement is met, a list of loadable forms will be returned for which the authenticated *TRAP user* has previously saved.
Persistent Change	--
User Satisfaction	5 (max)
User Dissatisfaction	5 (max)
Related Requirements	--
Conflicts	--
Support Materials	--
Test Cases	4.a

Requirement 4.b

Title	**Saved forms persist while *TRAP* is running**
Number	**4.b**
Type	Primary and Essential
Use Case	Save a form (#1)
Introduction	*TRAP* only ensures that form data will be saved while *TRAP* is running. If for any reason the *TRAP* system has to restart, any saved form data will be lost.
Rationale	This requirement is intended to simplify the design of the *TRAP* system.
Author	Brian Maurer, Andrew Helgeson, Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	**--**
Requirement Description	Any forms saved while *TRAP* is running should persist and be loadable by users. If *TRAP* stops running there is no requirement that form data is kept.
Outputs	--
Persistent Change	All form data will be lost after a restart or termination of the *TRAP system*.
User Satisfaction	1
User Dissatisfaction	1
Related Requirements	**--**
Conflicts	**--**
Support Materials	**--**
Test Cases	**4.b**

Requirement 4.c

Title	**Load a form**
Number	**4.c**
Type	Primary and Essential
Use Case	Load a Saved Form (#4)
Introduction	*TRAP shall allow users to load previously saved forms.*
Rationale	*TRAP users likely won't complete forms all at once or will want to save a template of a form. The TRAP system has a corresponding save function so it make sense to provide the functionality in the opposite direction, loading.*
Author	Brian Maurer, Andrew Helgeson, Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	A form id will be submitted by the current user when requesting a form.
Requirement Description	A user shall be able to load previously saved forms. If the submitted *form id* is invalid, in that there is no saved form with that id, *TRAP* shall return a *status error*.
Outputs	The form data and description saved under the *form id*. If the *form id* does not exist in *TRAP* a *status error* will be returned.
Persistent Change	--
User Satisfaction	5
User Dissatisfaction	5
Related Requirements	**--**
Conflicts	**--**
Support Materials	**--**
Test Cases	4.c

Requirement 4.d

Title	**Submit a form**
Number	**4.d**
Type	Primary and Essential
Use Case	Submit a Reimbursement Form to TRAP for Auditing (#5)
Introduction	The *TRAP GUI* shall be able to submit a *form* to the *TRAP system* for *auditing*.
Rationale	The *TRAP system* handles the *auditing* of *reimbursement forms*. In order to successfully do this, there must be a way to *submit* a *form*.
Author	Brian Maurer, Andrew Helgeson, Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	All of the form fields and form data
Requirement Description	All *required fields* must be filled out.
Outputs	Something to indicate successful submission The necessary deductions from the given grants
Persistent Change	Upon successful submission and audit, the results of the *reimbursement form* will be saved for later approval by an accountant.
User Satisfaction	5
User Dissatisfaction	5
Related Requirements	--
Conflicts	--
Support Materials	--
Test Cases	4.d

Requirement 4.e

Title	**Save a form**
Number	**4.e**
Type	Primary and Essential
Use Case	Save a form (#1)
Introduction	*TRAP* shall allow *users* to save a *reimbursement form*.
Rationale	*TRAP users* may want to save a form for many reasons including, to work on later or to save a template form to use as a starting point for future reimbursements. These are only a few motivating scenarios for this requirement.
Author	Andrew Helgeson
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	● The form data to be saved. Look at the *TRAP Input Dictionary* for information on these input fields. ● Only one of the following fields ○ A *form description* if the form is to be saved as a new form (ie a new *form id)* ○ A *form id* to save the form under. This will replace any data currently saved under this *form id*.
Requirement Description	A request can be to *TRAP* via the *diver interface* to save a form. This request requires the input specified above and it will be processed as follows. If there is a form description associated with the request, *TRAP* will create a new *form id* and save the *form data* under this new id. *TRAP* will then return a *status ok*. If there is a *form id* associated with the request, *TRAP* will overwrite the data of an already saved form with the given *form id* with the provided form data. If the *form id* does not exist in the *TRAP* system, the save request will be cancelled and *TRAP* shall return a *status error*. If both a *form id* and *form description* are provided for the save request *TRAP* shall reject the request and return a *status error*.
Outputs	*Status ok* - If the form data was successfully saved *Status error* - If the form data was not successfully saved
Persistent Change	The provided form data shall be saved under either a new *form id* or over the *form id* provided in the request. The saved form data should be immediately available for *form loading* after the save request has been successfully completed.
User Satisfaction	5
User Dissatisfaction	5
Related Requirements	**--**
Conflicts	**--**
Support Materials	**--**

Test Cases	4.e

Requirement 4.f

Title	**Remove all forms**
Number	**4.f**
Type	Primary and Essential
Use Case	Save a form (#1)
Introduction	*TRAP* shall allow *users* to remove all saved *reimbursement forms* and their metadata.
Rationale	A *TRAP user* may have collected a large amount of saved *reimbursement* forms and for organization may wish clear all forms.
Author	Andrew Helgeson
Source	Group Meeting, Friday Sept. 28, 2012 (Updated Nov. 4, 2012)
Inputs	
Requirement Description	A request can be made to *TRAP* via the *diver interface* to remove all saved forms and their metadata. The forms to remove are found by the *current user*. If a *current user* is not set in *TRAP*, a *status error* shall be returned. Otherwise, *TRAP* shall locate all saved form data and metadata for the *current user* and remove them from *TRAP*. After this operation, no forms shall be listed as loadable for the *current user* until a successful save form request is made some time later. As well, any attempts to save a form under an old *form id*, one which has been removed, shall fail with a *status error.*
Outputs	*Status ok* - If the *current user* is set and *TRAP* successfully removed all form data and metadata. *Status error* - If the *current user* is not set.
Persistent Change	All form data and description previously held under the *current user* will be removed and no longer available.
User Satisfaction	3
User Dissatisfaction	2
Related Requirements	4.e
Conflicts	**--**
Support Materials	**--**
Test Cases	4.f

Requirement 4.g

Title	**Set User**
Number	**4.g**
Type	Primary and Essential
Use Case	Set User (#5)
Introduction	A *TRAP user* must be set for various functionalities to work (see “Requirement Description” below).
Rationale	This is required for other functions to work.
Author	Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	• An *x500* corresponding to the *TRAP user*
Requirement Description	The provided *x500* must be looked up in the *userDB* for validation. If the *x500* is valid, the *current user* shall be set to the *x500.* Otherwise a *status error* shall be returned. An *x500* must be set for the *current user* in order for requirements 4.a - 4.f and use cases 1 - 4 to be valid.
Outputs	A *status error* shall be returned if the *current user* is not set (this applies for 4.a - 4.f). If an *x500* is valid, the *current user* is set and a *status ok* is returned, else a *status error* is returned.
Persistent Change	The *current user* is set to the provided x500 and the *current user* shall not change until it is re-set or the *TRAP system* is restarted.
User Satisfaction	5
User Dissatisfaction	5
Related Requirements	4.a-4.f
Conflicts	**--**
Support Materials	**--**
Test Cases	4.g

5. Post-Audit State

Requirement 5.a

Title	**Form output is available after a successful audit**
Number	5.a
Type	Primary and Essential
Use Case	Save a Form, Load a Form, Form Submission
Introduction	*TRAP* must be able to load the form output after successfully *auditing* a form.
Rationale	This is required functionality that *TRAP* must provide.
Author	Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	A *user* has *submitted* a *form* to *TRAP* for *auditing*. • <*form id*>
Requirement Description	If a *user submits* a *form* for *auditing*, the inputted <*form id*> shall be checked for existence. If the <*form id*> does not exist, a *status error* shall be returned. If the <*form id*> exists, *form auditing* shall continue. See requirement 5.d for handling the output of a successfully *audited form*. The output of a form shall be saved over the existing form associated with the <*form id*>.
Outputs	If a <*form id*> does not exist, a *status error* shall be returned. If a *form* has incorrectly formatted or validated fields, or if the *form* fails to conform to requirements 2.a - 2.n and 3.a - 3.f a *status error* shall be returned. On any *status error*, the original form shall remain unchanged and available for loading and saving. If a *form* is successfully *audited*, *TRAP* shall overwrite the form associated with the <*form id*> with the output.
Persistent Change	--
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	4.c - 4.e, 5.b, 5.d
Conflicts	--
Support Materials	--

Test Cases	5.a

Requirement 5.b

Title	**Original form still loadable/saveable after a *TRAP Error***
Number	5.b
Type	Primary and Essential
Use Case	Save a Form, Load a Form, Form Submission
Introduction	*TRAP* shall be able to load or save a form after a *TRAP Error*
Rationale	This is required functionality that *TRAP* must provide.
Author	Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	A *user* has *submitted* a *form* to *TRAP* for *auditing*. • *<form id>*
Requirement Description	If a form submitted for *auditing* has a *status error* returned, the form shall still be able to be loaded and saved.
Outputs	If a *<form id>* does not exist, a *status error* shall be returned. If a *form* has incorrectly formatted or validated fields, or if the *form* fails to conform to requirements 2.a - 2.n and 3.a - 3.f a *status error* shall be returned. The original form shall remain unmodified.
Persistent Change	--
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	4.c - 4.e, 5.a, 5.d
Conflicts	--
Support Materials	--
Test Cases	5.b

Requiremtn 5.c

Title	**Error Handling**
Number	5.c
Type	Primary and Essential
Use Case	Form Submission
Introduction	*TRAP* must be able to determine errors in a *form* and report these errors to the *user*.
Rationale	This is required functionality that *TRAP* must provide.
Author	Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	A *form* that has been *submitted* for *auditing*.
Requirement Description	Upon *submission* of a *form* for *auditing*, *TRAP* shall check the input fields for correct formatting and validation (requirements 1.a - 1.f) as well as conforming to requirements 2.a - 2.n and 3.a - 3.f. If any of these requirements are not met, a *status error* shall be returned. A *status error* shall report to the *user* the following: ● The input field causing the error ● A message that clearly describes (in human readable terms) the error ● If applicable, any help regarding the formatting of the field (e.g. a date field) *TRAP* shall only return the first error discovered; if there are multiple errors, they will be dealt with in later *form submissions* by the *user*.
Outputs	If an error is detected, a *status error* shall be returned.
Persistent Change	--
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	1.a - 1.f, 2.a - 2.n and 3.a - 3.f (these requirements relate to formatting and business logic)
Conflicts	--
Support Materials	--
Test Cases	5.c

Requirement 5.d

Title	**TRAP Output**
Number	5.d
Type	Primary and Essential
Use Case	Form Submission
Introduction	Upon successfully *auditing* a *form*, *TRAP* will generate output containing calculations based on the *audited form*.
Rationale	This is required functionality that *TRAP* must provide.
Author	Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	A *form* that has been *submitted* for *auditing*.
Requirement Description	If the *submitted form* does not meet requirement 5.b, a *status error* shall be returned. If a *form* passes *auditing*, the output in the *TRAP Output Dictionary* shall be filled out by *TRAP*. Additionally, if a *form* passes *auditing*, the output shall be saved internally within the *TRAP system* for later retrieval by the *user*.
Outputs	If a *form* does not pass *auditing*, a *status error* shall be returned. If a *form* passes *auditing*, the output shall be sent to the *driver interface* along with the output being saved within the *TRAP system* for later retrieval.
Persistent Change	--
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	4.a, 4.b, 4.d
Conflicts	--
Support Materials	--
Test Cases	5.d

Requirement 5.e

Title	**User Full Name Format**
Number	**5.e**
Type	Primary and Essential
Use Case	Submit a *Reimbursement Form* to *TRAP* for *Auditing* (#4)
Introduction	*TRAP* shall output a *name* field that is in the correct standard format. Check the *TRAP Output Dictionary* for other information on this output.
Rationale	There is a standard format that is used for people's names and *TRAP* shall obey these formatting conventions.
Author	Andrew Helgeson
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	• A lastname • A firstname • Optional: name_suffix • Optional: middle initial/name
Requirement Description	The outputted name should be formatted as such. With entries inside [ ] being optional and < > meaning required: <lastname> [name_suffix], <firstname> [middle] This input information is gathered using the x500 login of the *TRAP user* with the *userDB*. See the "World Assumptions" section for more information on the *userDB*.
Outputs	A name formatted as shown above, <lastname> [name_suffix], <firstname> [middle]
Persistent Change	--
User Satisfaction	5 (max)
User Dissatisfaction	5 (max)
Related Requirements	5.f - name information is gathered using this x500
Conflicts	--
Support Materials	--
Test Cases	5.e

Requirement 5.f

Title	**Presentation**
Number	5.f
Type	Primary and Essential
Use Case	Form Submission
Introduction	If a *user* presents at a conference, they must report the title, abstract of work and show they mentioned their grant sources.
Rationale	This is required functionality that *TRAP* must provide.
Author	Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	● <*Justification Presented*> ○ Yes or No ● <*Justification Presentation Title*> ● <*Justification Presentation Abstract*> ● <*Justification Presentation Acknowledgement*>
Requirement Description	If a *user* is presenting at a conference, input field <*Justification Presented*> shall be set to “yes”. The remaining input fields must be present, otherwise a *status error* shall be returned. The remaining input fields must also have data (not empty), otherwise a *status error* shall be returned.
Outputs	● <*Presentation*> ● <*Presentation Title*> ● <*Presentation Abstract*> ● <*Presentation Acknowledgement*>
Persistent Change	--
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	--
Conflicts	--
Support Materials	--
Test Cases	5.f

Requirement 5.g

Title	**Username**
Number	5.g
Type	Primary and Essential
Use Case	Form Submission
Introduction	A <*username*> will be included with the *TRAP* output.
Rationale	This is required functionality that *TRAP* must provide.
Author	Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	● <*Username*> ○ x500 of the *user*
Requirement Description	The <*Username*> shall be included in the *TRAP* output. If not present in the input, a *status error* shall be returned. See the "World Assumptions" regarding the proper authentication of a *user*.
Outputs	● <*Username*>
Persistent Change	--
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	--
Conflicts	--
Support Materials	--
Test Cases	5.g

Requirement 5.h

Title	**Visa Status**
Number	5.h
Type	Primary and Essential
Use Case	Form Submission
Introduction	The visa status of a *user* will be included in the *TRAP* output.
Rationale	This is required functionality that *TRAP* must provide.
Author	Ethan Waytas
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	● <*Username*>
Requirement Description	The visa status shall be included in the output. The *userDB* shall be queried using the <*Username*>. If no results are returned (*database timeout*), a *status error* shall be returned.
Outputs	● <*Visa Status*>
Persistent Change	--
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	3.f (vista status used in making grant restriction decisions)
Conflicts	--
Support Materials	--
Test Cases	5.h

6. Non-Functional Requirements

Requirement 6.a

Title	**Database Timeouts**
Number	6.a
Type	Non-Functional and Necessary
Use Case	Non-Functional requirement. Applies for any *TRAP* use case.
Introduction	It should be expected that the database systems which *TRAP* communicates with may encounter delays or full disconnectivity. *TRAP* should react in a consistent and predictable manner in these situations.
Rationale	It would be naive to assume databases or any other component in the system will have 100% uptime and perfect response. With this in mind it is important to define what should happen in these scenarios so the customer is not startled and the system's response is reliably testable.
Author	Andrew Helgeson
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	The non-functional requirement is not dependent upon *TRAP user* input.
Requirement Description	Through each of the use cases specified in this requirements document, *TRAP* must interact with some form of persistent storage, which is likely a database. This is especially evident during *form auditing* where grant information, currency rates, per diem amounts and so on must be looked up in order to finish the operation. In the case of database delay or failure, the *TRAP* system shall wait no longer than the *database timeout* for a response. After the *database timeout* has elapsed, the current request to the *TRAP* system shall terminate and a *status error* shall be returned.
Outputs	*Status ok* - If there are no problems with the databases, all database requests return within the *database timeout* and the request to *TRAP* completes successfully. *Status error* - If any database request exceeds the *database timeout*. While not part of this requirement you could receive a *status error* if some other part of the request to *TRAP* fails.
Persistent Change	If the pending database request would have mutated the state of the database it should be ensure that even after timeout and *status error* return the database state is as it was before the call. This mainly applies where communication is delayed.
User Satisfaction	1
User Dissatisfaction	5 (Max)
Related Requirements	--
Conflicts	--
Support Materials	--
Test Cases	6.a

Requirement 6.b

Title	**Business logic modifications**
Number	6.b
Type	Non-Functional and Necessary
Use Case	None
Introduction	*Business logic* will change throughout the life of *TRAP* so it shall be efficient with respect to time and money to add, remove or modify *business logic.*
Rationale	Through *TRAP*'s deployment lifetime, *business logic* will change so for the customer it is essential that these *policies* be easy to modify.
Author	Andrew Helgeson
Source	Group Meeting, Friday Sept. 28, 2012
Inputs	● The *TRAP* system ● A *business logic* to add, remove or modify
Requirement Description	Addition, modification or removal or *business logic* in the *TRAP* system should take no longer than 1 hour for an *average developer*. If *business logic* policies are viewed as a *module,* the 1 hour constraint for a developer to add the logic does not include the time to actually write the logic *module*. This time only references the time it should take the developer to add the logic *module* into the *TRAP* system so it becomes enforceable.
Outputs	A *TRAP* system with appropriately modified *business logic.*
Persistent Change	The persistent change is in the *TRAP* system. Its main function, *form processing*, is now altered by new *business logic.*
User Satisfaction	5 (Max)
User Dissatisfaction	5 (Max)
Related Requirements	--
Conflicts	--
Support Materials	--
Test Cases	6.b