

Crash Magic Online

Crash Records Analysis



Crash Magic Online

Online Crash Records Analysis Software

Pd' Programming, Inc.

Crash Magic Online is the browser-based version of Crash Magic.

Crash Magic provides an excellent interface to analysis of your crash records database.

Crash Magic Online

The Crash Magic software and accompanying documentation is the property of Pd' Programming, Inc. and is protected by both United States Copyright Law and international treaty provisions. Please refer to your license statement for specific license information.

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Licensed users of this software with a current subscription may obtain written permission to reproduce this document by contacting Pd' Programming or logging into our web site. (www.pdmagic.com)

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

This manual, as well as the Crash Magic software is copyrighted material.

Copyright 2003-2024, Pd' Programming, Inc. Lafayette CO. All rights reserved.

Publisher

Pd' Programming, Inc.

Managing Editor

Pete d'Oronzio

Technical Editors

Scott Miller

Casey Vincent

Cover Graphic

Scott Knaur

*Printed: November 2024 in
Lafayette, CO USA*

Table of Contents

Foreword	0
I. Introduction	1
Overview	2
Using the program	3
User roles	4
Technical support	5
Help system and manual	6
Finding what you're looking for	6
II. Tutorial	8
Overview	9
Log in	9
Your project home page	10
Help	11
Create your first study	12
Create your first collision diagram	15
Select a schematic	16
Change the text labels	18
Click On information	20
Other diagram settings	23
Create your first chart	24
Change the chart field	25
Change the sort order	29
Other chart settings	31
Create your first crash list	32
Sorting the list	35
Copy and Paste	35
Save and load templates	36
Use the filter	38
Printing	41
Log out	42
III. How do I	43
Questions about logging in	44
How do I create a shortcut or favorite to Crash Magic?	44
How do I start the Crash Magic program?	44
Is my login or password case sensitive?	44
My login or password isn't working	44
Questions about projects	45
How do I create a new project?	45

How do I delete an existing project?	45
Questions about studies	46
How can I export my crash records	46
Questions about reporting	46
How can I create a report button	46
Questions about collision diagrams	47
How can I change the curb lines	47
Questions about lists	47
Questions about charts	47
How can I change the fields displayed	47
Questions about filters	47
How can I share my filter	47
Questions about users	48
How do I add a user	48
 IV. Reference	 49
Add in utilities	50
cm Map Magic for ArcGIS	50
cm Local Viewer	50
cm Node Mapper	50
Aliases	50
Street name aliases	50
Intersection aliases	59
Node aliases	61
Duplicate intersections	61
Category List	62
Category list selector	62
Category list editor	63
Common interface elements	65
Logout, help and home buttons	65
New report buttons	65
Common toolbar buttons	66
Crash Magic Local Viewer	67
LocalFile	67
LocalExe	68
Security	69
Data Entry	69
Adding streets	69
Exporting data	74
Export study form	75
Exporting to XLSX using a template	75
Filters	76
Filter Selector	77
Edit form	78
Operators	79
Functions	80
Examples	86
Logging in	87

Login form	87
Projects	90
Project home page	90
Project settings tab	91
User settings tab	92
Environment tab	93
Reports	94
Charts	96
Collision diagrams	99
Crash listings	103
Cross Tab	108
High Crash Locations	112
Layouts	121
Studies	136
Study settings tab	137
Study info tab	139
Study types	140
Templates	143
Load template	143
Save template	145
V. Administration	146
Overview	147
Data	147
Installation	147
1. Project manager steps	148
2. Server administrator steps	149
3. Database administrator steps	168
4. Group administrator steps	172
Reference	182
Data Entry	182
Database	185
Help system	193
Login Methods	194
Maintenance Form	200
Multiple program instances	209
PSRattrs	210
Query Editor	297
System Panel	301
The program folders	302
User	304
User Group	305
Additional roadway systems	307
Importing SYS configuration data	307
Troubleshooting	314
Debug url parameters	314
Launch sequence and timing	314
Process log	315
VI. Utilities	316

PdForXML.exe	317
PdForXML - Examples	318
VII. Prepare a new configuration	321
Overview	322
Resources	322
Required and Recommended fields	323
Required data for a configuration	325
Configuration checklist	326
Tools and utilities	329
Obtain data and metadata	330
User questions	330
Prepare to start the configuration	330
Create and Prepare Users and Groups	331
Create the User Group	331
Create and update users	332
Create the .config User	332
Create the .shared User	333
Log out. Log in.	333
Create common PSRattrs	333
.Options - General	333
Raster image	333
Connection	333
SQL script	334
Text lines	334
Import - DB to XML	334
Import - XML to DB	335
Locale info	335
Create and populate crash database	335
Create database and tables	335
Prepare a sample source file	336
Convert source data to XML	336
Configuring Corridor Diagrams	336
temporary	336
Configuring High Crash Location reports	337
Step By Step	338
Distinct intersection query	338
HCL query	344
VIII. Automation and interoperability	345
Overview	346
Deprecated	347
Basic URL	347
MagicAuto	349
Using GET or POST	349
Actions	350
Remote Access	363
Data structures	364
IcmRA_Admin	365

IcmRA_Analysis	376
IcmRA_Login	385
Languages	390
Version log	392

Index	393
--------------	------------



Chapter

I

1 Introduction

1.1 Overview

Crash Magic (CM) is a graphic display and data summary package designed for use in safety management systems for analysis of crash data. CM is interactive, offering detailed collision diagrams, data retrieval, crash summaries, statistical output, and user specified graphic displays. CM provides traffic safety specialists and law enforcement officials an exceptional tool for identifying crash patterns, high crash locations, and maintenance and operational concerns.



The most unique feature of CM is its configurability. CM's diagrams and graphics can be changed to suit what your specific preferences are. From changing the diagram schematic to a T Intersection, to adding comments, or creating your own charts and configuring them to the limits of your imagination.

With CM's filtering features, ad-hoc queries can be used to create a particular diagram or chart, or can be used to create specific reports. For example, you can run a report showing where the High Crash Locations are for all Pedestrian crashes that occurred in the rain. Filters can be created and combined using all the fields in your database!

Another great feature of CM is its interactive capabilities. CM allows the user to point to individual crashes and review the crash record information on screen, and crashes may be dragged to new locations with annotations added. This feature eliminates constant referrals to hard copy reports to explain unusual patterns or other specific data.

Crash Magic is designed for easy access to traffic crash data. It is capable of generating hundreds of collision diagrams, charts, pin-maps, and listings in the time it would take a technician to generate one. It therefore has a wide range of applications in

any setting where officials are concerned with identifying correctable crash patterns. CM generates standard collision diagrams, and provides highlighting and customized selection (filtering) of user specified crash types. For example, if left-turn crashes are of particular interest, the user can highlight these crashes on the screen and on the printout. By selecting left-turn crashes in the filter, they will be the only type displayed.

Crash Magic can diagram individual intersections of particular interest, corridors of intersections along a primary street, or large numbers of collision diagrams for record keeping or statistical purposes. Each graphic printout includes the collision diagram, area name, crash count, user-defined text, and the city name.

Crash Magic allows easy identification of high crash locations and recurring crash patterns for use in safety and hazard elimination programs and for site specific review of proposed developments or geometric changes. Operationally, CM can be used to identify correctable traffic control and intersection lay-out problems. CM offers immediate access to crash information for quick response to citizen concerns by allowing access to intersection data and easy access to individual crash reports that may be in question.

Diagrams and other reports are easily and quickly generated for any study. Buttons at the top of the project tree provide access to create the most common reports.

Studies determine the data to be included in a report. All the reports created within (below) a study, contain a representation of the crashes in the study. This means that changing the settings of a study will affect all of its reports. One powerful setting of all studies is the filter. A filter can be used to limit the crashes included in a study.

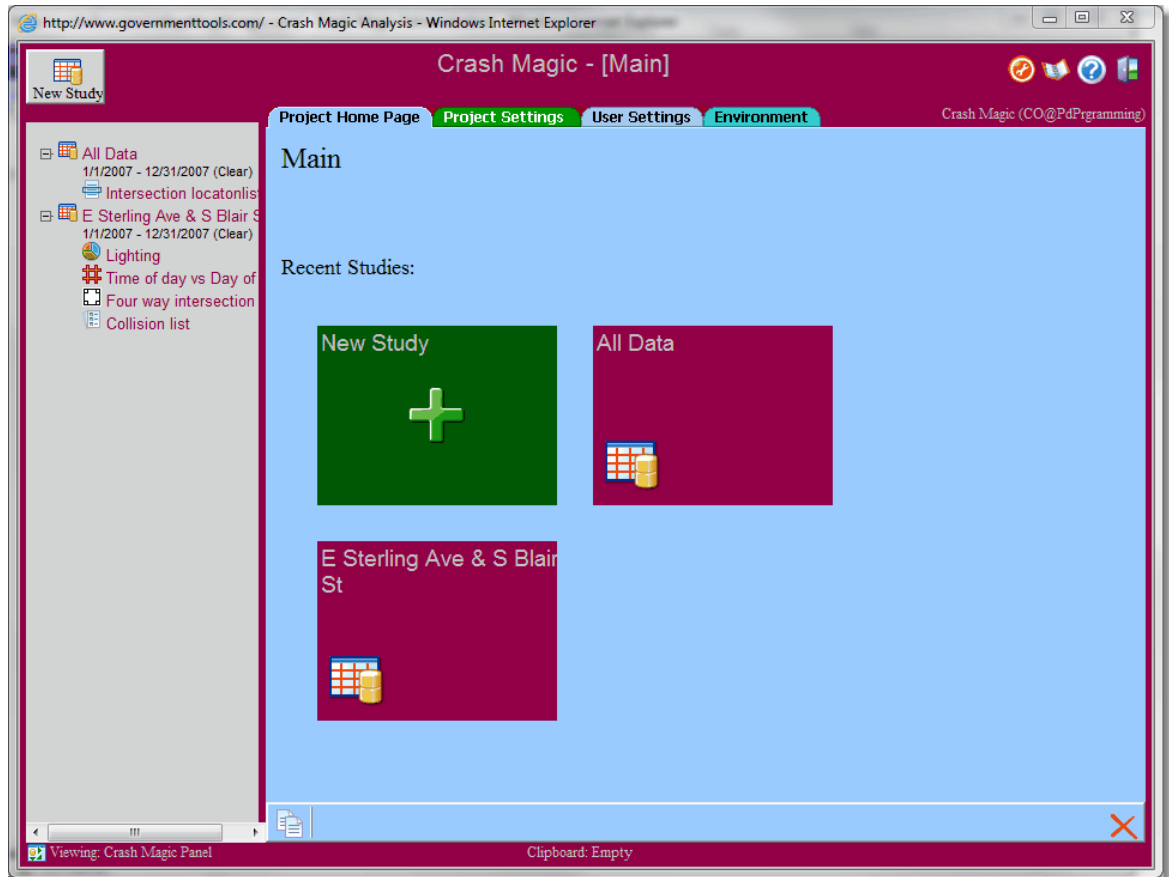
- Diagrams can be manipulated by dragging crashes or adding symbols or descriptive text.
- A listing of the crashes can be generated for a given query. The standard listing is a report of all the crashes including any fields chosen from the database.
- Charts of any type can be generated based upon the fields in the database to give a graphical look at a study.
- A high crash location list can be generated on systems that have specified street names, mileposts or geographic coordinates.
- Pin maps can be generated for most systems that can produce high crash location lists.

1.2 Using the program

Crash Magic's main form includes several elements.

On the left side of the window is the "Project tree" which contains all the datasets, referred to as "studies", and reports that have been generated over time.

The main part of the screen, on the right side, contains the report content. This is where diagrams, listings, charts and other reports will be displayed. Tabs above report content area provide access to report settings and other options.



There is no need to explicitly "save" your work in Crash Magic. Every move is saved as you navigate amongst the many available reports. By using the program's copy and paste features, reports can be replicated and modified to consider many "what if" scenarios.

User roles

Depending on your organization, you may have database administrators, system administrators, project managers, etc. Each of these people has played an important part in preparing Crash Magic for your use. Once they've completed their job and the system is running, there are only two or three roles you'll need to worry about:

- **Analyst** - This is the role most users of the software will fall into. The analyst has access to all the data gathering and report generating parts of the program. Analysts only have the ability to consume the data, not to modify it. An analyst need not worry about accidentally modifying data, breaking the software, or making any changes that

will affect other analysts. The effects of their actions only affect their own workspace, and can be easily returned to the defaults if a mistake is made.

- Data entry - If your agency is using Crash Magic for data entry, you will have one or more people with logins that permit them to enter the data entry portion of the program.
- Group administrator - This role is a very important one. The group administrator is an analyst who has also been given the ability to modify the system configuration and to manage the sharing of data and templates across other analysts.

As an analyst using the program, it is important to know who your Crash Magic group administrator is.

1.3 Technical support

Pd' Programming provides excellent technical support. Our support professionals are available during normal business hours, (Mountain Standard Time) 5 days a week. With the exception of long distance charges, technical support services are free. We'll do everything we can to help you understand Crash Magic better and make full use of its capabilities.

To contact Pd' Programming you can call, write, fax, or send email. The best way to contact Pd' Programming is by email. However, when contacting us for telephone support, please have the following information available:

- Program version number (Available on the login form).
- Microsoft Windows version number.
- Browser name and version
- Any unusual messages from the program.

The program has several functions that can provide much more useful information to us when problems arise. Therefore, if possible, please be seated at your computer with the program running when you call.

Address: Pd' Programming, Inc.
725 Aegean Drive
Lafayette, CO 80026

Telephone: (303) 666-7896

Fax: (303) 666-7347

EEmail: help@pdmagic.com

Web site: www.pdmagic.com

File uploads: To send a file to Pd' Programming, please log into our website and use the "upload" option on the right side of the page. This will ensure that we are notified of the upload and will streamline the process for you.

File downloads

If Pd' Programming has a file for you, it will be made available when you log in and select the "downloads" link on the right side of the page.

1.4 Help system and manual

Crash Magic documentation is thorough. The documentation is divided in to sections for analysts ("tutorial", "how do I...", and "Reference"); for administrators; and for programmers. (Automation and interoperability) When the help button is clicked in Crash Magic, the list of topics presented is always context-sensitive, showing information based on where you are in the program. It is therefore helpful to take as many steps as you know before clicking the help button for assistance.

In addition to the online help system, viewable in a browser, the help system is available in a number of formats:

- Adobe Acrobat (PDF) for computer viewing, searching and printing
- Windows html help for computer viewing with advanced navigation
- Windows e-book format for computer and tablet viewing
- HTML format for online viewing
- Printed format, based on the Adobe Acrobat file

All of the various help media contain the same content. The latest version of the content in all formats is always be available on the PdMagic.com web site.

1.5 Finding what you're looking for

It's time to dive in! Where you start depends on your current level of knowledge of the program and what you are interested in learning.

As you use the various sections of the manual, you will find that many topics reference other topics in the manual. When using an electronic version of the manual, these references are "hot links" and can be clicked on. The printed version, as well as the

PDF designed for printing will include page numbers like this  instead.

A great place to start if you've never used Crash Magic is the "Tutorial" section.

The tutorial is a step-by-step guide to some of the more popular features of the program.

It will guide you through each click and key press to create data and reports and to print to share them. It describes the basics of changing some report settings, some nifty features of the program to make your work easier, and most importantly, it provides links to more detail about each of the features in case you need additional information.

If you have a grasp of how the program works, but are having a hard time figuring out how to do a specific task, the "How do I.." section provides suggestions from the very highest to low levels of the program. The focus of this section is translating the name you'd choose for a feature into the name we've chosen. For example, the text next to each crash? The crash annotations? The description? The date and time? We call those the "labels". By looking in the "How do I.." section, you'll find out what the correct Crash Magic terminology is, and find links to the sections in the manual that can help you figure out how to accomplish what you're working on.

The reference section is for when you're staring at some settings, or a bunch of buttons, or a report, and just want to know all the options you have. Perhaps you're just curious, or perhaps you know what you want but can't seem to find the control that will do that for you. This section contains the detail that would bog down the Tutorial, and clutter up the "How do I" section.



2 Tutorial

2.1 Overview

Welcome to the Crash Magic Tutorial section. These tutorials are designed to help you become familiar with the basic functionality of the Crash Magic software.

The purpose of each tutorial is to walk you through the steps to perform a specific task. There are often several methods to accomplish a task in the program, but this tutorial will choose one method. There are often a number of options available when performing a task. Again, this tutorial will address a few, but not all of these options. Throughout the tutorial, you will find links to the Reference section of the manual where details about the current task can be found. After completing the tutorial, you will be in a good position to investigate the rest of the program, try other options, and utilize the rest of the manual to learn the complete power of the program.

The tutorial contains descriptions, references and instructions. Instructions are steps to be followed as you move through the tutorial. ***They will be highlighted like this.***

Please be sure to read the [Introduction](#) ² section of the manual before proceeding to the tutorial.

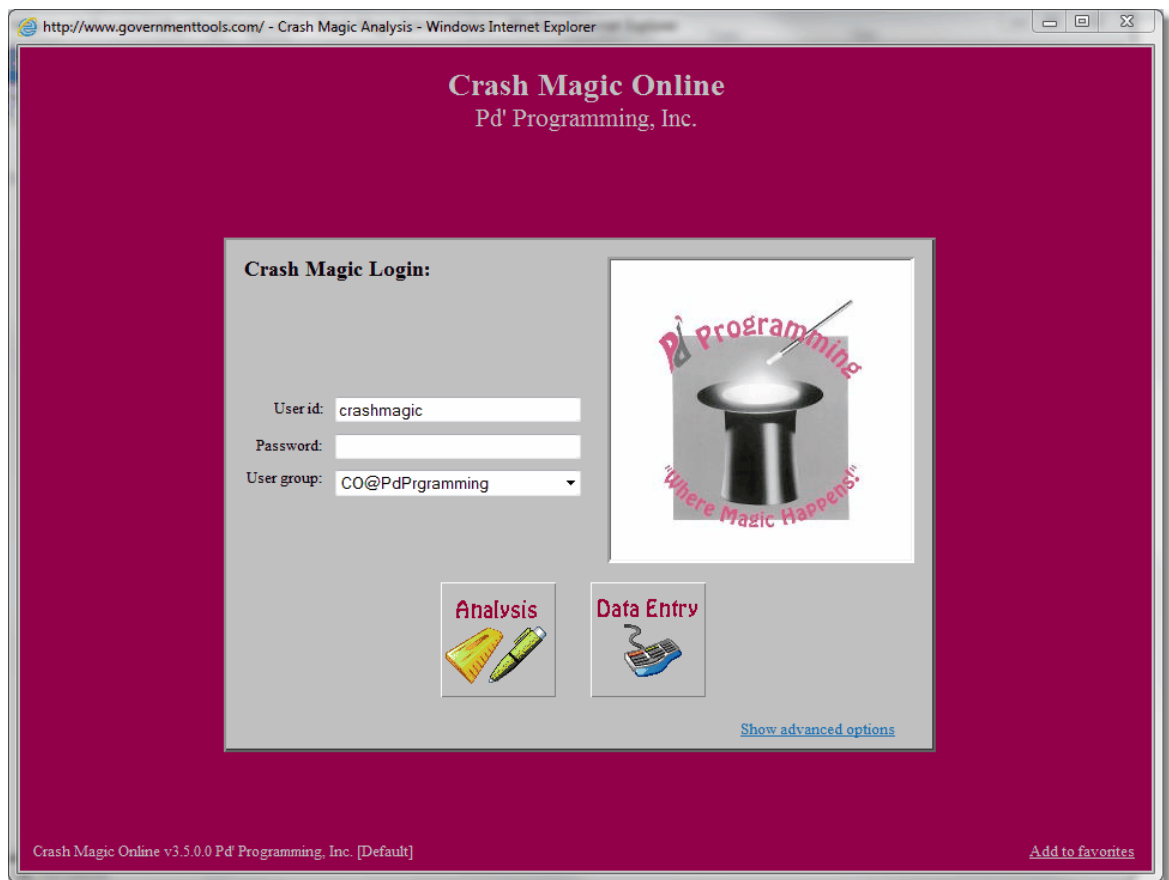
One note: Crash Magic is an ever-evolving piece of software providing new functionality and options all the time. In addition, Windows and your browser are being updated and enhanced as well. We've done our best to present images of the screens as you should see them, but they may not always appear exactly as they are in the manual. In the event that something doesn't match exactly, please feel free to inform us, but don't let it stop you from continuing your exploration of the program. Chances are the option you are looking for is still there and, once selected, will get you back to matching screens.

If Crash Magic is installed on your jurisdictions server, the first step is to ***obtain the url where the software is installed.*** Your project manager or system administrator should have this information. If Pd' Programming is hosting Crash Magic for you, then proceed to www.governmenttools.com, and hit the Log In button.

2.2 Log in

Click on the link or enter the url to enter the Crash Magic software application.

This is the login form:



Your administrator should provide you with a login, password, and your user group name. **Enter this information and click "Analysis".**

Depending on the installation of Crash Magic some clients may be automatically logged into the analysis side of the program when the url is opened.

See also: [Login panel](#)^[87], [Questions about logging in](#)^[44]

2.3 Your project home page


After logging in, you will be presented with your project home page. This screen provides access to several important functions of the program. From here, you can access help; create new projects, studies and reports; and you've got a link back to the last study you were on when before you last logged off.

You'll notice that the Crash Magic program is divided into two sections. The "project tree" on the left, and the report content on the right. The project tree is available throughout the application. It allows quick access to data and reports that have been created in your project. You can easily move between reports by clicking on them in the tree. The report section on the right is the content that you are currently viewing. In this case, it is the project home page.



See also: [Projects](#)^[90], [Questions about projects](#)^[45]

2.4 Help

All of the screens in Crash Magic include a help button  in the top right corner of the page. **Click this button now.**

Crash Magic will open the Help Navigator. As you use the program, the help system is always in the background "watching" where you go. When you request help, the Help Navigator presents you with help topics based on the forms you've navigated to this point. The first topic in the list is the most specific, with each item below becoming less specific, down to the last item "Crash Magic Online" which is the most general. As you move about the program, items are added and removed as needed to provide the most context sensitive help possible.

The image below is what you will see in the Help Navigator when you click on it from the Home Page. "Home Page", the most specific topic, is at the top; with the more general "Crash Magic Online" below it.

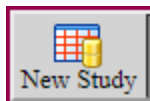


Click the link to "Tutorial overview". You are presented with a new browser displaying the Crash Magic documentation. **Browse the documentation a bit and then close the Help Navigator to return to Crash Magic.**

Note: The help system may be customized with content specific to your agency. See [customizing the help system](#)¹⁹³.

2.5 Create your first study

In order to generate a report, you must first decide on the data you would like the report to display. Perhaps you are looking for all the crashes at a particular intersection for the past year. Or, you might be looking for the crashes that occurred on a particular stretch of highway. Depending on how you would like to gather your data, you will need to select a particular "study type". **Start by clicking on the "New study" button above the project tree.**



The study settings panel will be displayed for you to select your desired study type and pick criteria for it to use while gathering data. For example, an intersection study will require a date range and two street names. A route milepost study will require a date range, route name and starting and ending milepost. **From the 'Type' drop down list, choose an intersection, node or route-milepost section with a sampling of crashes available.**

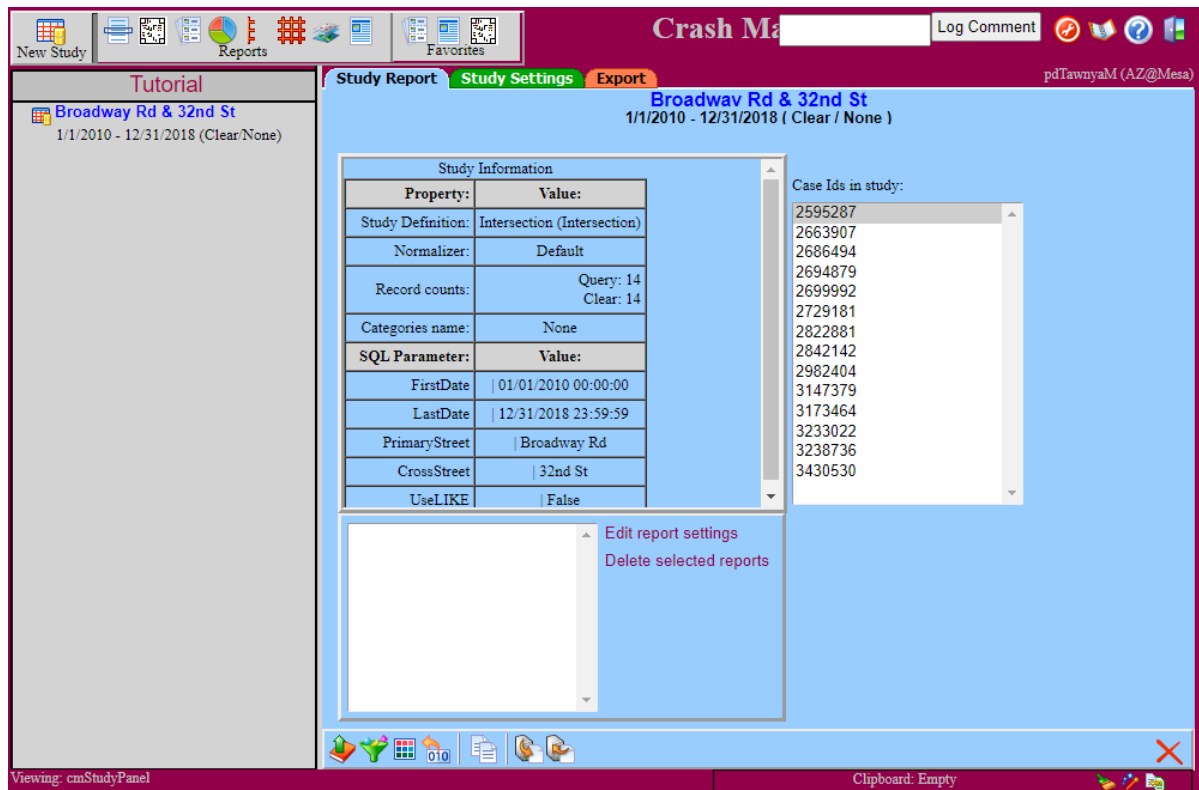
Note about available study types: Crash Magic supports many types of studies. Your installation of the software has been customized to show templates for studies that can be supported by your data. For example, as a local agency with no rural roadways, you may not have a "milepost" field. In that case, your configuration will not include a "route milepost" study template. However, you are likely to have an

"Intersection" or "Address" study template. Possible study types are, Date Range (by date), Case Id (list by number), Intersection (by 2 streets), Route Milepost (by street and milepost), Address (by street and address), X Y Rectangle (by Latitude and Longitude), and Node (by node number).

Depending on your choice of study type, you will now be presented with several properties to fill in such as street names, dates, etc. **Fill in those properties.** Once you have everything filled in, you may hit the blue update arrow to show how many crashes that you have within the study you just created. If you've got values (other than 0), you're good to go. If not, return to the settings and change the criteria so that the study contains data. (i.e. different dates, streets, id's, routes, etc.) A common problem is a time period without data, so be sure to include a valid date range.

The screenshot displays the 'Crash Map' application interface. The top navigation bar includes 'New Study', 'Reports', and 'Favorites' buttons. The main content area is divided into a left sidebar and a right panel. The sidebar, titled 'Tutorial', shows a study named 'Broadway Rd & 32nd St' with a date range of '1/1/2010 - 12/31/2018 (Clear/None)'. The right panel, titled 'Study Settings', contains several input fields: 'Name' (Broadway Rd & 32nd St), 'Description' (empty), 'Type' (Intersection), 'Filter' (Clear), 'Categories' (None), 'Date range' (1/1/2010 thru 12/31/2018), and 'Intersection' (Broadway Rd and 32nd St). A blue update arrow is visible next to the 'Filter' field. The bottom status bar indicates 'Viewing: cmStudyPanel' and 'Clipboard: Empty'.

You'll notice that the page you are on contains several tabs. The one you are currently working in is the green settings tab. **Click the blue "Study Report" tab.** You will now be presented with a report showing you some basic information about the data you have selected.



For the purposes of this tutorial, it is best to have a query with 5 to 75 crashes in it.

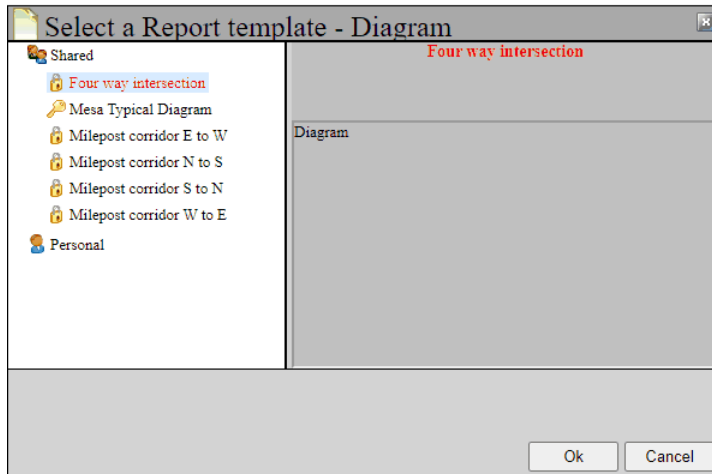
Now you have your first study. Every study and report within that study are saved automatically. So next time you open the program, they will be there for you. At this point, you can always click/activate the study from the tree on the left, and make any changes desired in the study on the green tab.

Note about using your own safety management system to gather data: In some agencies, the data selection is done by your own in-house system. Your safety management system has a link, or button that you can click on to "send" the data to Crash Magic. In this case, you will not click on a study template button for this tutorial step, but rather, **you should now log into your safety management system and request that it send a selection of crashes to Crash Magic.** Please see your internal documentation for more information on this step. Depending on how your system was written, you may jump to the end of this step, or all the way to a new diagram or chart.

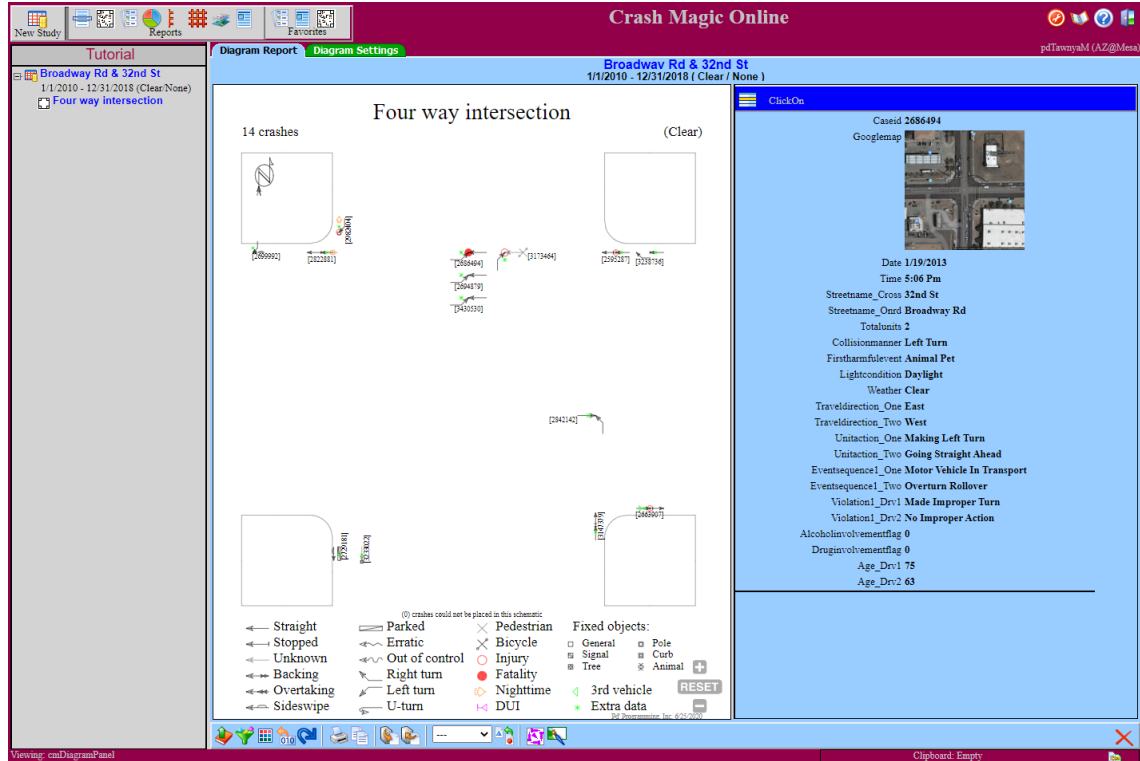
See also: [Studies](#)¹³⁶, [Questions about studies](#)⁴⁶

2.6 Create your first collision diagram

Now let's create a collision diagram. **Click on the diagram button**  **at the top of the project tree.** You will be presented with a few of the basic schematics available for diagrams. **Choose the Four way intersection option and hit OK.**



A collision diagram report will be created and displayed in the main part of the screen.



Congratulations! Your first diagram in Crash Magic.

The title is displayed at the top of the diagram. Any filters in use on the study will be displayed under the title. A legend for the crash symbols will be displayed at the bottom of the schematic. The blue section to the right of the diagram will display any detail information that you have selected for crashes that you have selected. More about [Click On information](#)^[20] later in the tutorial.

There are many options for changing the appearance of a collision diagram and the data in the click-on report. We will cover a sampling of this in the tutorial.


See also: [Collision diagrams](#)^[99], [Questions about collision diagrams](#)^[47]

Select a schematic

Crash Magic uses a "[schematic](#)"^[102] to assign each crash symbol a location, and to display the appropriate curb lines. Schematics are available to represent a variety of intersection and corridor types. Schematics are selected from the settings tab. **Click on the green "Diagram Settings" tab at the top of the diagram now.**

The settings tab contains a number of options for formatting your collision diagram. One of those settings is the schematic selection. The schematic that you want to use depends on the type of location you have specified in your study. If you have selected data from an intersection, you can choose from a variety of intersection schematic types. If you have selected data gathered from a corridor, you should choose a corridor schematic.

If you gathered data for an intersection:

1. **Open the list of available schematics by clicking on the  and select the "Int_4/1_Turn/E_Bound" schematic.** This schematic represents a 4 leg intersection with 1 turn lane in the East-bound direction. **Now click on the "Diagram Report" tab** to return to the diagram. The new diagram will depict the intersection using the selected schematic.
2. Now, **go back to the Diagram Settings tab and change the schematic to "Int_4/nNormal"**, the default schematic. **Click on the Diagram Report tab** to return to the collision diagram.

If you gathered data using a Route-milepost study:

1. **Open the list of available schematics and select the "MilepostEW" schematic.** This schematic represents a corridor where the mileposts increase as they proceed East to West. **Now click on the "Diagram Report" tab** to return to the diagram. The diagram is now displayed as a corridor with crashes grouped into areas based on how far along that corridor they occurred.
2. Now, **go back to the Diagram Settings tab and change the schematic to match the data you requested.** This will be one of the Milepost?? options. **Click on the Diagram Report tab** to return to the collision diagram.

If you gathered data by another method, you will need to identify an appropriate schematic to render that data. In cases where your data is gathered in a unique manner, you may have some custom schematics. For example:

Oregon DOT:

1. **Open the list of available schematics and choose the "OR/INT_4/N_E_S_W" schematic.** These schematics were created specifically to address the DOT's data which contains specific direction information for each of the streets. (up to 8 possible directions for each leg) **Now click on the "Diagram Report" tab** to return to the diagram. The diagram is now displayed as a simple 4 leg intersection.
2. Now, **go back to the Diagram Settings tab and change the schematic to match the data you requested.** This will be one of the OR/INT_?/??? options. **Click on the Diagram Report tab** to return to the collision diagram.

See also: [Schematics](#) 


Change the text labels

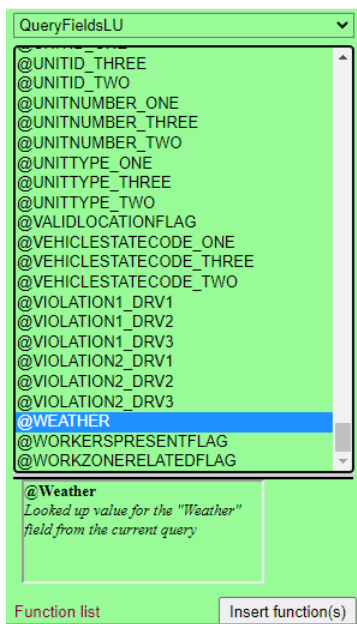
As you view the diagram, note that each crash has some text next to it. The default label should resemble your case id field. However, the labels may be configured to include any fields in the database.



Select the Diagram Settings tab again and note the section for "Label settings". It includes options to set the color, content and size of the labels.



Click the  button next to the content box. This will open the Expression editor for crash labels. The program is capable of producing labels that combine fields, are conditional on other fields, and may include other complex content. For now, locate the combo box in the top right corner of the form.




Select the category called "QueryFieldsLU". (This may already be selected for you.) The LU stands for "looked up" meaning that the program will display English descriptions of the data rather than the codes in your database. These fields should look familiar, as they come from your own database. Scroll down the list and find the one that describes the weather in your database. **Select it and press the "Insert function(s)" button.** The text (something like "@Weather") will appear in the editor on the left.

Select the "Save and return" button.

While we're here, change the size of the labels to "25".

Now click the blue Diagram Report tab to show the diagram with your new labels.

Note: you can use the zoom control on the bottom right of the diagram, or your mouse wheel to zoom in and out. You can also drag the diagram to pan up, down, right or left.

Now to set the label back to the default case id number, **Select the Diagram Settings tab** again and **change the font back to "10"**. Also, **click the  button** to edit the labels. This time, **clear anything that is in the editor by either selecting the text and deleting it, or hitting the 'clear expression' button on the bottom.**

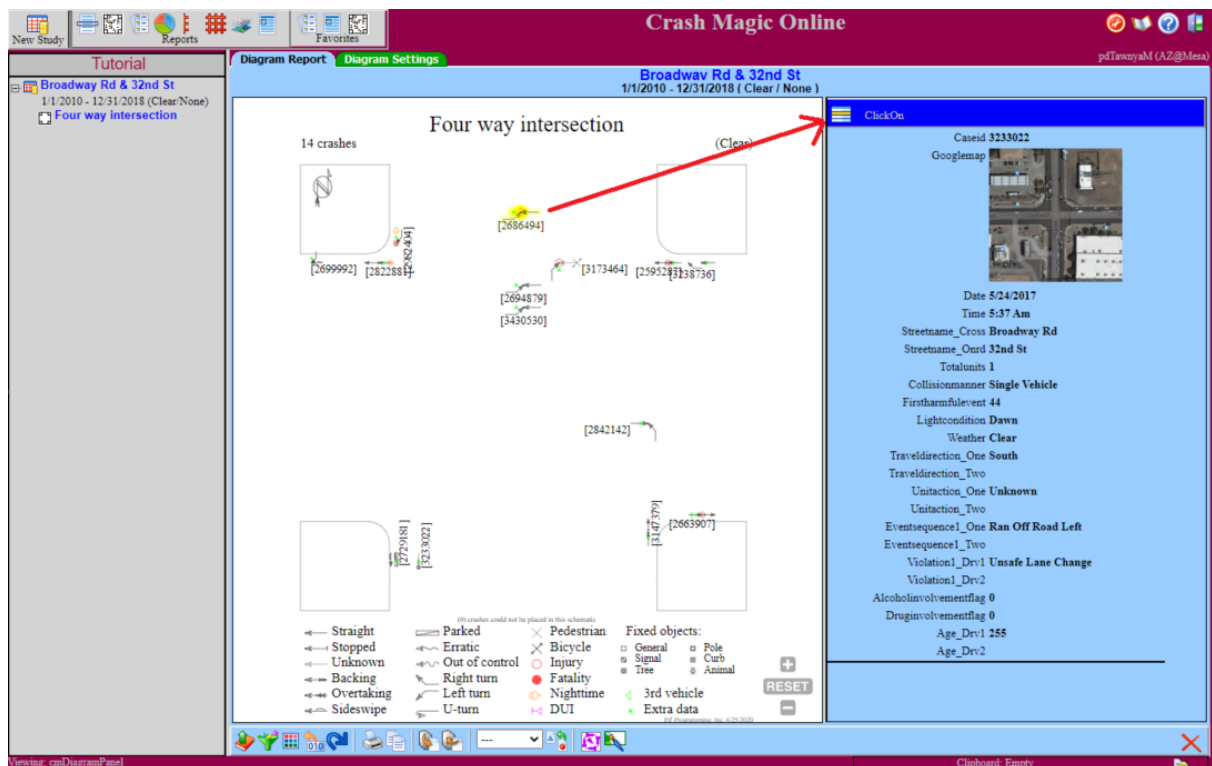
Select the "Save and return" button.

Now click the blue Diagram Report tab to show the diagram with the default label.

Click On information

One of the powerful features of the program is that you can click on a crash graphic and display the data that is in the database for that crash. As you move your mouse over each of the crashes, it will highlight in yellow. **Click a crash graphic now.**

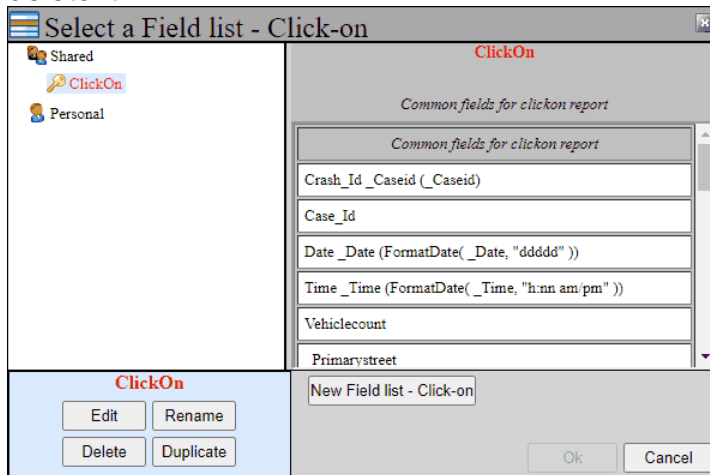
If this is the first time you've clicked on a crash, there probably haven't been any fields selected yet. In this case, the area to the right will be blank. To select or change the fields that are displayed when you click on a graphic, select the field list button in that far right panel.



Click the field list button now (shown above with arrow)

In several places throughout the program, you will be presented with a dialog that allows you to choose from saved field lists, charts, filters, or category lists. The first time you open this pick list, you will only have the shared Clickon available. Since this is an administrative resource (key shown), you may not be able to edit it. So down below you

can choose to 'Duplicate' the click on, which will make a copy of it into your personal directory. **Click the Duplicate button now.** Now you can edit this listing, rename it or delete it.



Click the Edit button now.

This will present the field list editor. This lists all the fields that are available in the database on the left side. The selected fields for the clickon report in the middle, and some advanced settings on the right.

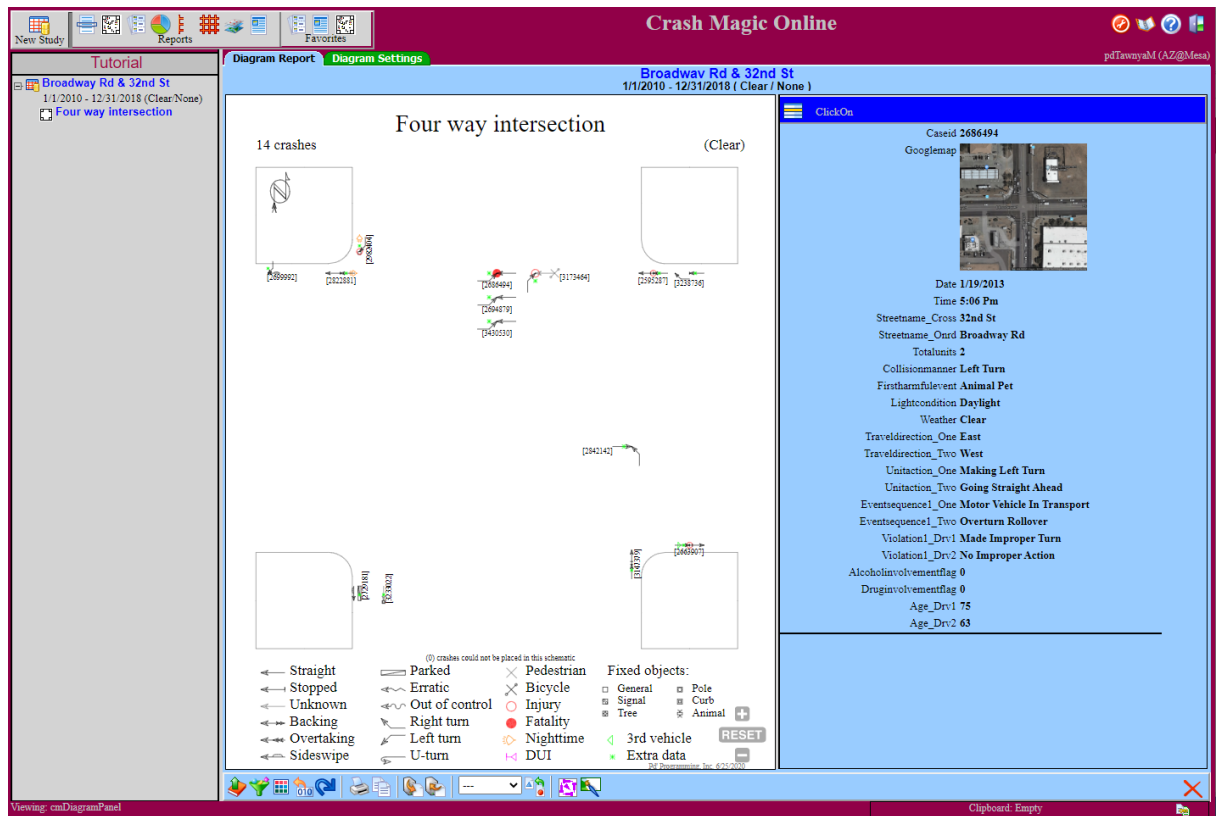
Crash Magic - Field List Editor

Name: ClickOn copy Description: Common fields for clickon report

Available fields:	Selected fields:	Field title:
QueryFields	Crash_Id	Crash_Id
ACTIVE_SCHOOL_ZONE_FLAG	Case_Id	Field description:
AMENDMENT_FLAG	Date	_Caseid
AT_INTRSCN_FLAG	Time	Expression:
CALCFATALITIES	Vehiclecount	_Caseid
CALCNONINCAPINJURIES	_Primarystreet	
CALCNONINJURIES	_Crossstreet	
CALCPOSSIBLEINJURIES	_Blockaddress	
CALCSUSPSEIOUSINJURIES	_Milepost	
CALCTOTALPERSONS	At_Intrscn_Flag	
CASE_ID	_Dirfromint	
CMV_10000LBS_OR_MORE_FLAG	_Distfromint	<input type="checkbox"/> Advanced
CMV_10000LBS_OR_MORE_FLAG	First_Harmful_Event	
CMV_10000LBS_OR_MORE_FLAG	Manner_Of_Collision	
CMV_BUS_TYPE_ID_V1	Total_Injury_Count	
CMV_BUS_TYPE_ID_V2	Veh_Action_V1	
CMV_BUS_TYPE_ID_V3	Veh_Action_V2	
CMV_CARGO_BODY_ID_V1	Veh_Direction_Of_Travel_V1	
CMV_CARGO_BODY_ID_V2	Veh_Direction_Of_Travel_V2	
CMV_CARGO_BODY_ID_V3	Light_Condition	
CMV_CARRIER_ID_TYPE_V1	Weather_Condition	
CMV_CARRIER_ID_TYPE_V2	Traffic_Control	
CMV_CARRIER_ID_TYPE_V3	Person_Age_D1	
CMV_CARRIER_ZIP_V1	Person_Age_D2	
CMV_CARRIER_ZIP_V2	Person_Charges_D1	
CMV_CARRIER_ZIP_V3	Person_Charges_D2	
CMV_DISABLING_DMG_FLAG_V1	Unit_Description_V1	
CMV_DISABLING_DMG_FLAG_V2	Unit_Description_V2	
CMV_DISABLING_DMG_FLAG_V3		
CMV_EVT1_ID_V1		
CMV_EVT1_ID_V2		
CMV_EVT1_ID_V3		
CMV_EVT2_ID_V1		
CMV_EVT2_ID_V2		
CMV_EVT2_ID_V3		
CMV_EVT3_ID_V1		
CMV_EVT3_ID_V2		
CMV_EVT3_ID_V3		
Active_School_Zone_Flag		
Access the "Active_School_Zone_Flag" field from the current query		

Cancel and return
Save and return

In the field list editor, you can select fields from the "available fields" list by clicking on them. (Ctrl+Click or Shift+Click to select multiple fields) **Select a dozen or so fields** and then **click the plus button**. This will add the fields to the "selected fields" list for your report. If you would like to remove a field from the selection, select it, then click the minus button, you can also move any selected field up and down in the list by using the arrow buttons. That's it. **Click "Save and return"** and then **"OK"** to the list selector to see the results.



This is now your default list of fields that will be displayed anytime you click on a crash in the program.


Other diagram settings

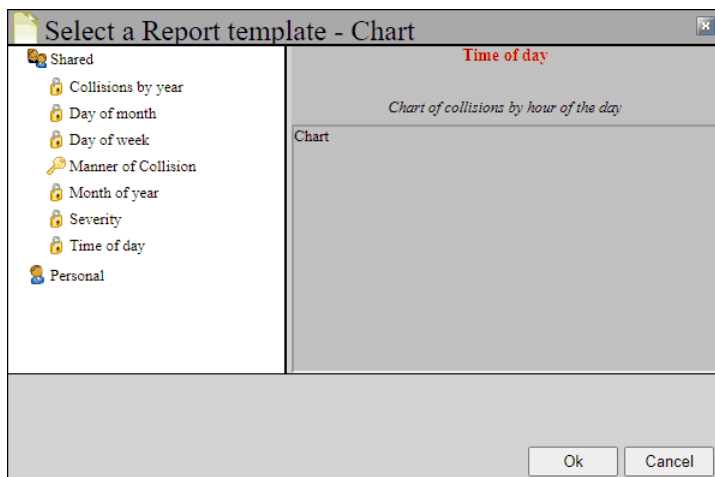
There are a number of ways to modify the diagram display. Many of them are available on the "Diagram Settings" tab.

Let's say you have a diagram with only a few crashes on it, and you want to annotate it or simply spread the crashes out so that they each have more space. **Click on the *Diagram Settings* tab.** "Spacing" is the distance between each of the crash graphics. The default spacing is 50. **Click on the number 50 and change it to 200.** While we're at it, let's also **uncheck the "Display legend" checkbox.** This will remove the legend from the bottom of the page.

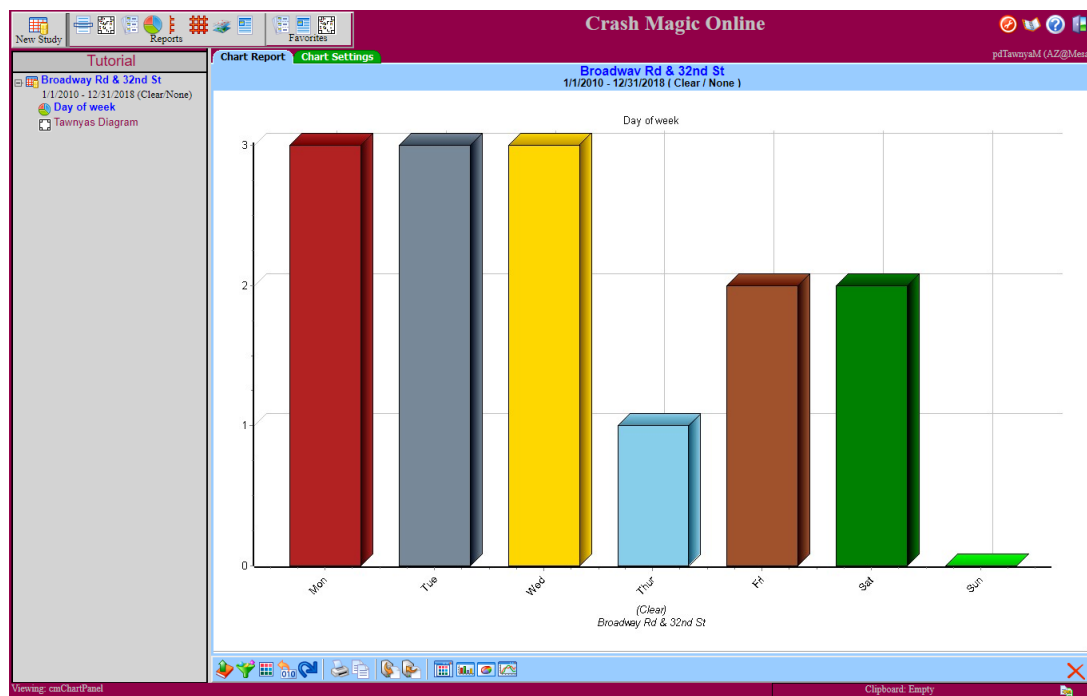
Click on the *Diagram Report* tab to return to the diagram and see your changes.

2.7 Create your first chart

Another way of displaying data is using a chart. Charts can be created using any field in the database. Let's create a chart. To do this, **click the chart button**  at the top of the project tree. This will bring up a list box to select a chart from. There are several charts already prepared for you under the Shared directory. **Choose the *Day of Week* chart,** and hit the **OK** button.



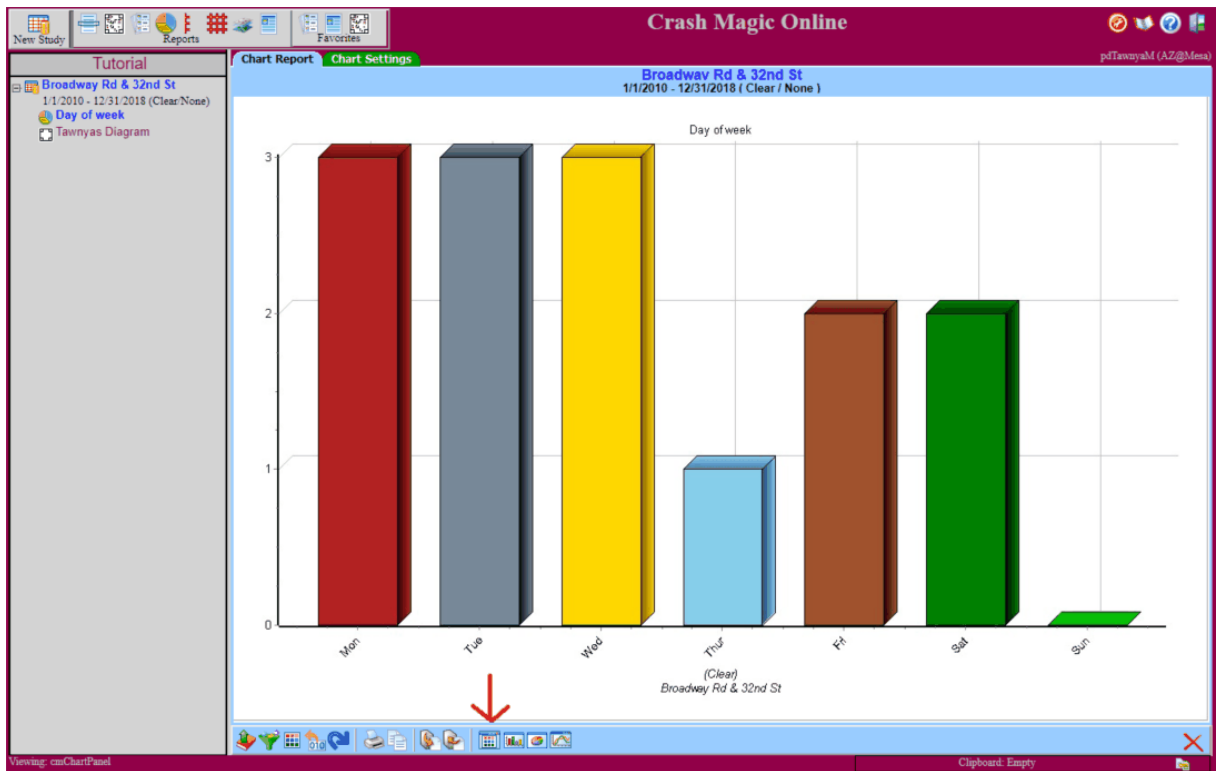
This will bring up a Day of Week chart for the study that you've been working in.




See also: [Charts](#)⁹⁶, [Questions about charts](#)⁴⁷

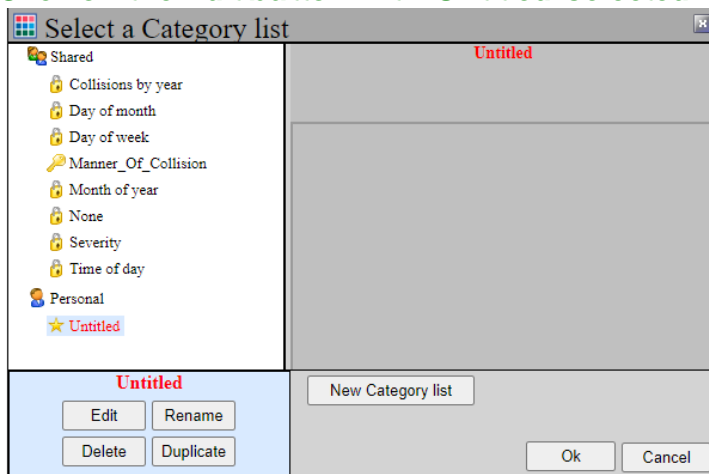
Change the chart field

To tell the program what we'd like charted, you can change or create a new category list.



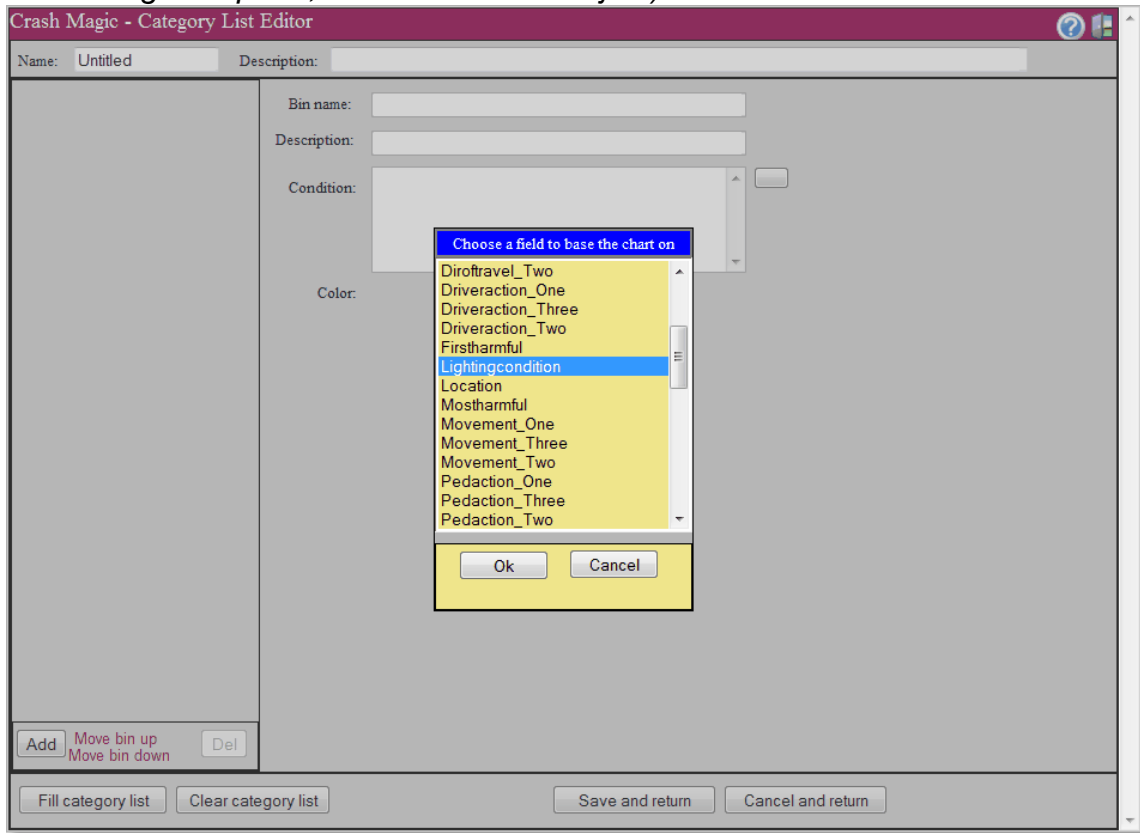
To change the chart category list while viewing a chart, **Click the far right category list button on the bottom of the page** . Here you can select an existing category list to build a chart from or Create a new list.

In this example we will create a new category list. **Click the New Category list button.** An 'Untitled' name will appear under your Personal account that can be edited. **Click on the Edit button with 'Untitled' selected.**

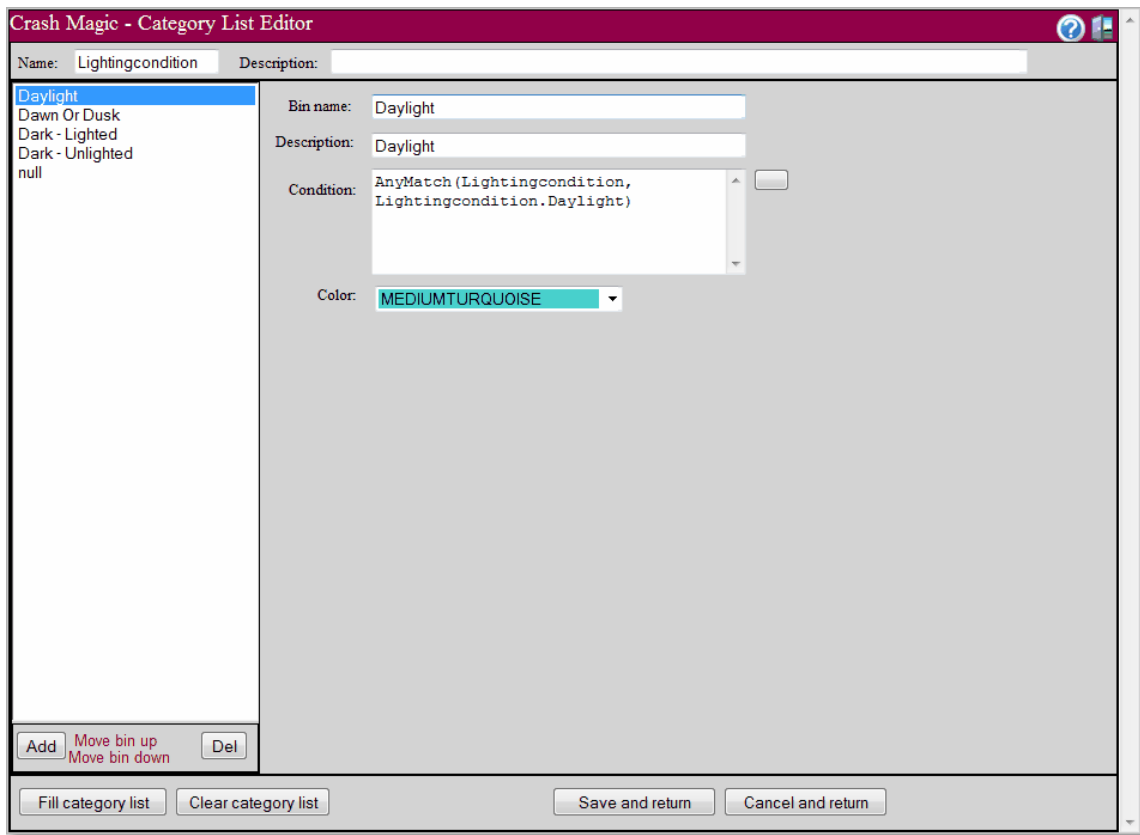


Here we can define a category from almost any of the fields in the database. **Click the button on the lower left to "Fill category list". Choose the field that indicates your lighting condition.** If you don't have that, choose your weather field. (we're

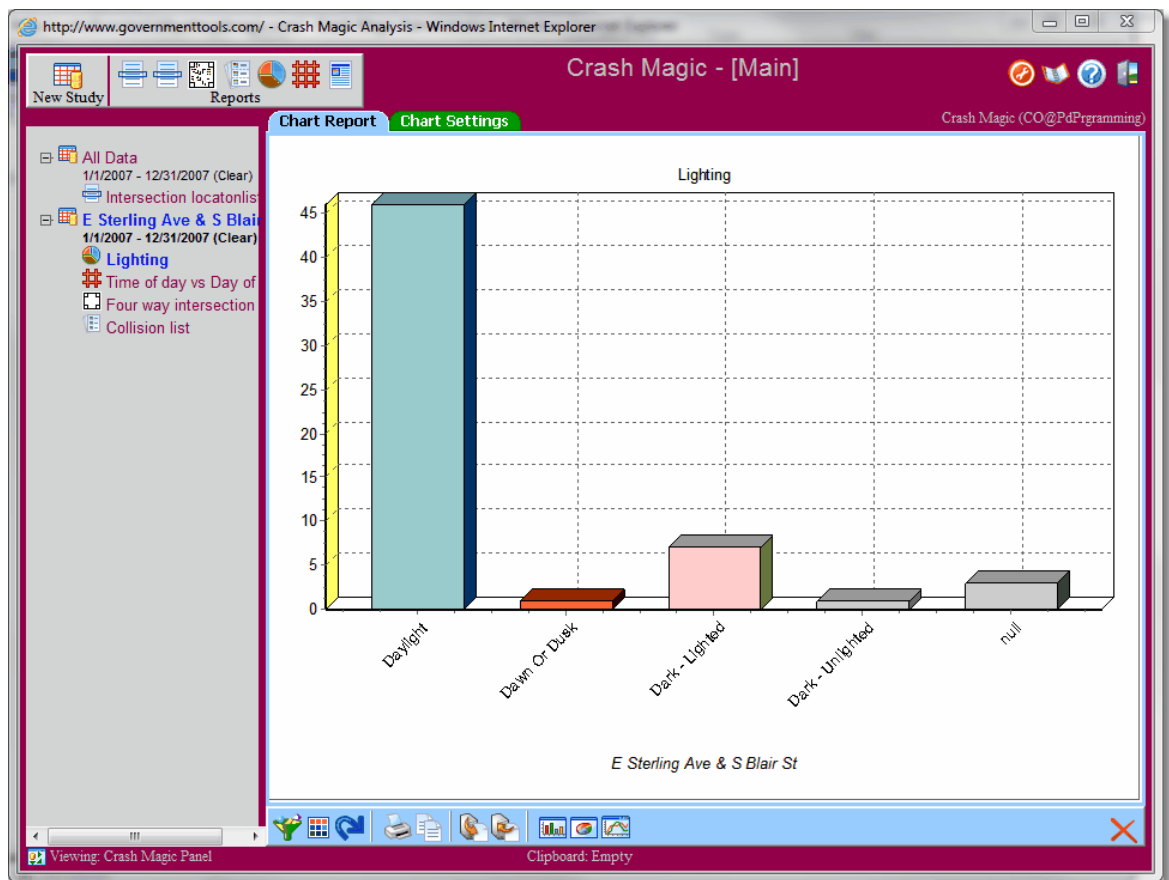
intentionally choosing "categorical" fields in this step. Charting "continuous" data, such as age or speed, is an advanced subject)



Click the *Ok* button. This will populate default "bins" for this field. "Bins" are Crash Magic's term for the containers that hold each category of data. They will be represented as bars, pie slices, or lines. The name of the database field is populated as the name of the Category list.

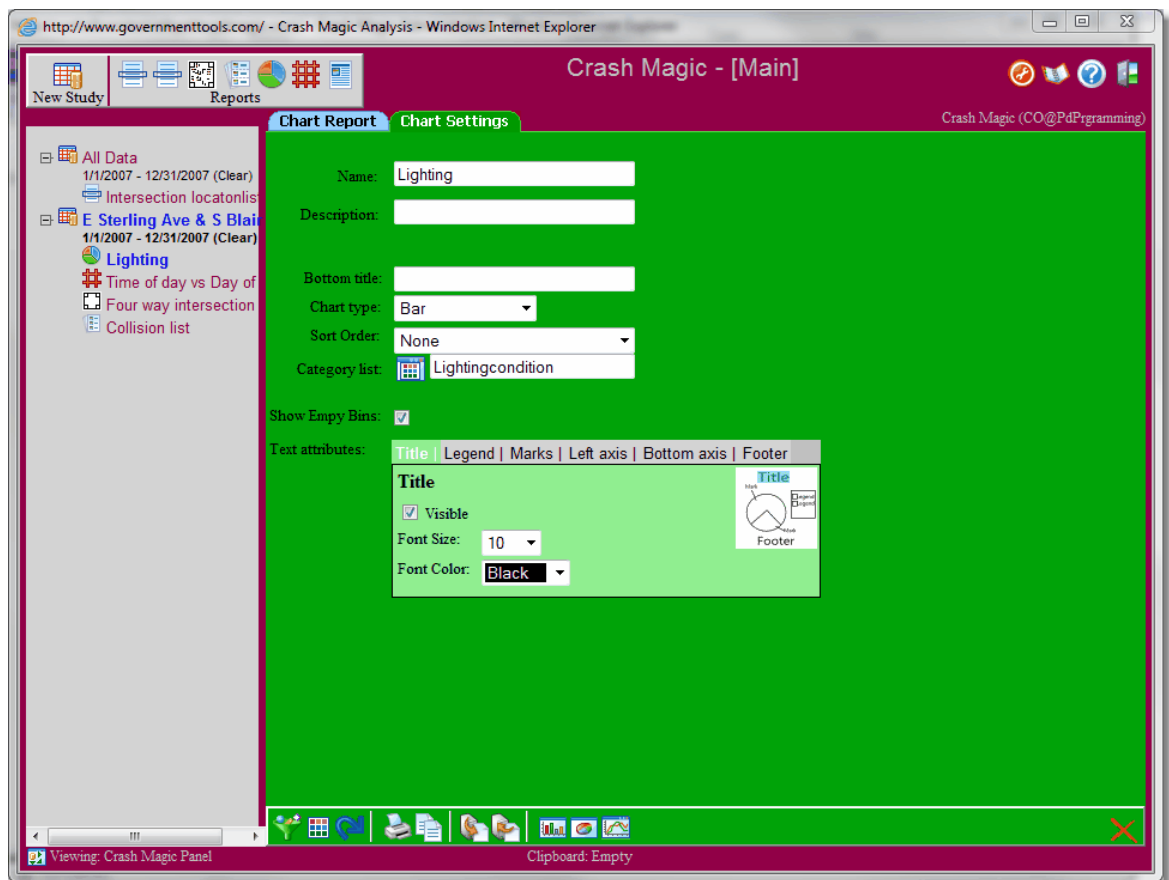


Click the "Save and return" button to return to the category list selector. The new category list should be selected by default, if you would like you can click on the rename button and give it a different name like 'Lighting' or 'Weather'. **Click the OK button** to close the selector, and your chart will be displayed.



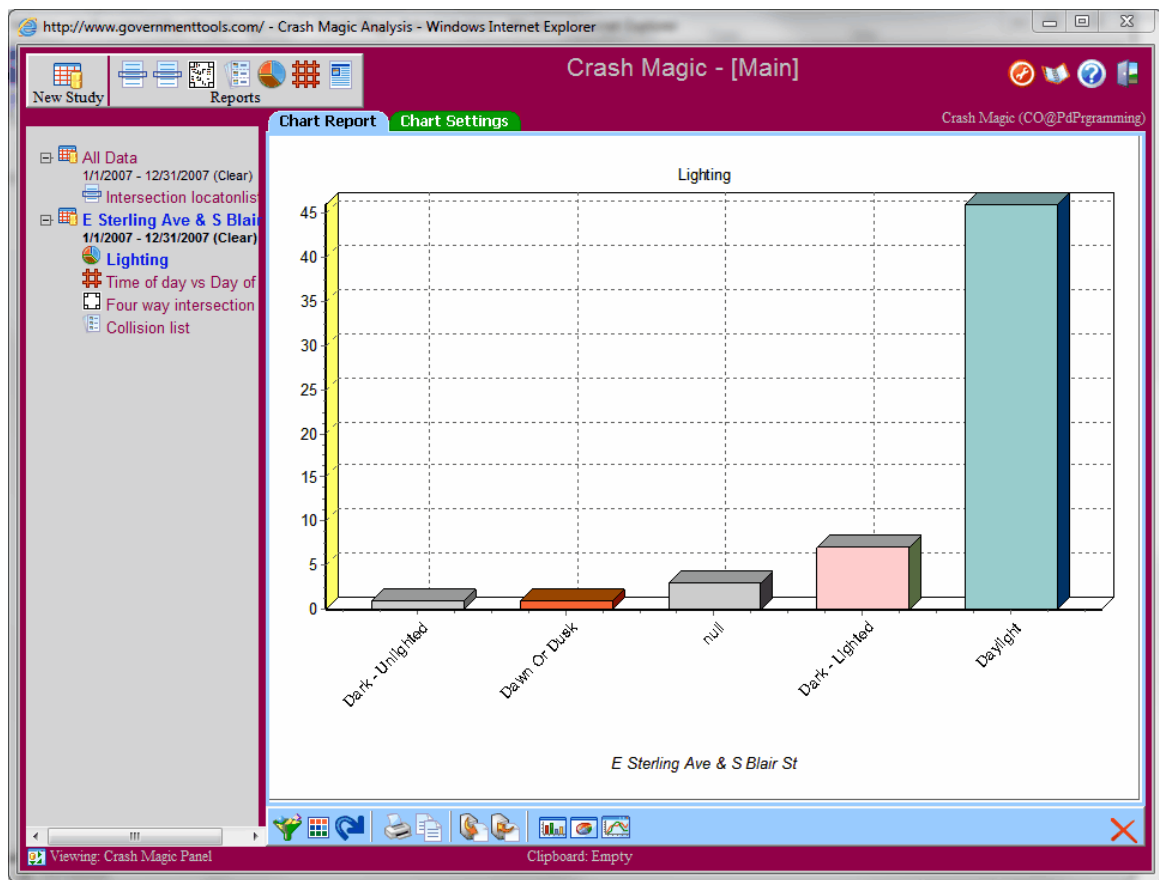
Change the sort order

As with the diagram, there are a number of settings that you can play with on the "Chart Settings" tab. Click the **Chart Settings** tab now. You can re-order the bins by selecting a different sort order. **Change the sort order to "Count"**.



You can also choose to remove empty bins from the chart by removing the check from the 'Show Empty Bins'. This can be handy if you have a lot of variables for a particular field.

Now, click the **Chart Report tab**. You can see that the sorting for the chart has now ordered itself from fewest crashes - to most crashes.



Other chart settings

There are quite a number of settings available to change the appearance of charts generated by Crash Magic. Let's try some of them out.

Click on the *Chart Settings* tab. Note the "Text attributes" section at the bottom of the form. This section contains properties used to show, hide or edit almost every facet of the text in the chart.

Select the "Title" tab and change the font color to Purple.
Select the "Legend" tab and choose "Visible".

Click on the *Chart Report* tab and view the new chart. A legend will now be visible with all the values in the chart. The text of those values will be Purple.


Some settings are used more than others and we've placed them at the bottom of the diagram in its own toolbar.



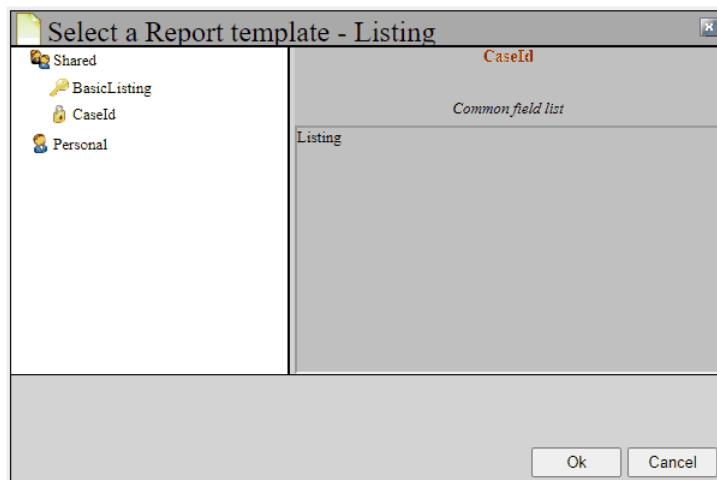
The most obvious are the three chart types: Bar, Pie, Line. **Select the Pie chart by clicking the button.** The chart will be rendered as a pie chart. This pie chart will have a legend, but will not show the value of each slice. To do this, we have to turn "marks" on under the settings tab.

Select the Chart Settings tab and **choose the "marks" tab** in the Text attributes section. Note the graphic on the right that highlights the part of the chart you are about to edit. **Check the "visible" option.** Now **return to the Chart Report** to see the slices with their new labels.

2.8 Create your first crash list

How about creating a nice basic list of the crashes at this location? To create a listing, **select the listing button** at the top of the project tree. 

You will be presented with the template selection for a field listing report. You may only have one available.

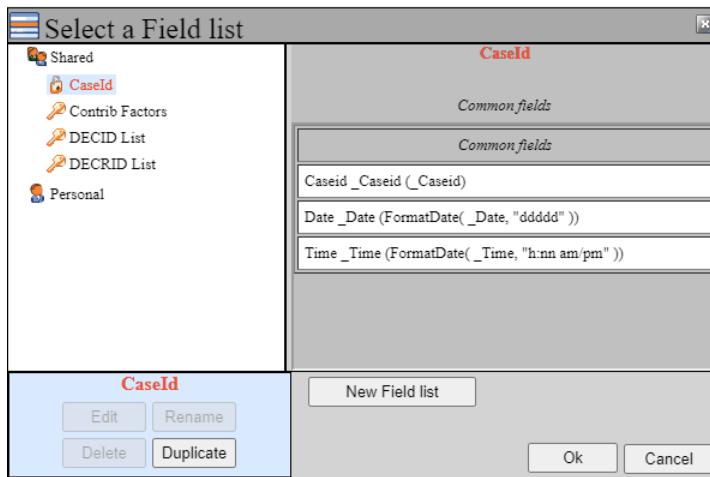


If you only have **CaseId** available, choose that one. And hit the **OK** button. You will get a field listing with only the case id, date, and time shown. In order to make your own field list, **now choose the field list button on the bottom of the screen.**



This will bring up the field list selection box. In several places throughout the program, you will be presented with a dialog that allows you to choose from saved field lists,

charts, filters, or category lists. The first time you open this pick list, you will only have the shared Caseld available. Since this is an administrative resource (key shown), you may not be able to edit it. So, down below you can choose to 'Duplicate' the Caseld list, which will make a copy of it into your personal directory. Or you can choose to create a new field list. **Click the New Field list button now.**



To specify fields to show, **click the "Edit field list" button** in the bottom toolbar.

Crash Magic - Field List Editor

Name: ClickOn copy Description: Common fields for clickon report

Available fields:	Selected fields:	Field title:
QueryFields	Crash_Id	Crash_Id
ACTIVE_SCHOOL_ZONE_FLAG	Case_Id	Field description:
AMENDMENT_FLAG	Date	_Caseid
AT_INTRSCN_FLAG	Time	Expression:
CALCFATALITIES	Vehiclecount	_Caseid
CALCNONINCAPINJURIES	_Primarystreet	
CALCNONINJURIES	_Crossstreet	
CALCPOSSIBLEINJURIES	_Blockaddress	
CALCSUSPSEIOUSINJURIES	_Milepost	
CALCTOTALPERSONS	At_Intrscn_Flag	
CASE_ID	_Dirfromint	
CMV_10000LBS_OR_MORE_FLAG	_Distfromint	<input type="checkbox"/> Advanced
CMV_10000LBS_OR_MORE_FLAG	First_Harmful_Event	
CMV_10000LBS_OR_MORE_FLAG	Manner_Of_Collision	
CMV_BUS_TYPE_ID_V1	Total_Injury_Count	
CMV_BUS_TYPE_ID_V2	Veh_Action_V1	
CMV_BUS_TYPE_ID_V3	Veh_Action_V2	
CMV_CARGO_BODY_ID_V1	Veh_Direction_Of_Travel_V1	
CMV_CARGO_BODY_ID_V2	Veh_Direction_Of_Travel_V2	
CMV_CARGO_BODY_ID_V3	Light_Condition	
CMV_CARRIER_ID_TYPE_V1	Weather_Condition	
CMV_CARRIER_ID_TYPE_V2	Traffic_Control	
CMV_CARRIER_ID_TYPE_V3	Person_Age_D1	
CMV_CARRIER_ZIP_V1	Person_Age_D2	
CMV_CARRIER_ZIP_V2	Person_Charges_D1	
CMV_CARRIER_ZIP_V3	Person_Charges_D2	
CMV_DISABLING_DMG_FLAG_V1	Unit_Description_V1	
CMV_DISABLING_DMG_FLAG_V2	Unit_Description_V2	
CMV_DISABLING_DMG_FLAG_V3		
CMV_EVT1_ID_V1		
CMV_EVT1_ID_V2		
CMV_EVT1_ID_V3		
CMV_EVT2_ID_V1		
CMV_EVT2_ID_V2		
CMV_EVT2_ID_V3		
CMV_EVT3_ID_V1		
CMV_EVT3_ID_V2		
CMV_EVT3_ID_V3		
Active_School_Zone_Flag		
Access the "Active_School_Zone_Flag" field from the current query		

Cancel and return

Save and return

This form should look familiar, it's the same editor used for the click on report in the diagram panel. **Select a few fields**, pressing the **plus** button to move them to the "selected fields" list. **Click the "Save and return" button**. Then the **OK** button from the pick list.

CaseId
Filter: Clear

Date	Time	Int	Vehicles	Fatal	Injuries	Manner	First_Harmful	Contrib_Factor1_Id_V1
1/15/2010	10:16 am	No	1	No	0	ONE MOTOR VEHICLE - GOING STRAIGHT	FIXED OBJECT	FAULTY EVASIVE ACTION
1/23/2010	1:47 pm	No	2	No	0	SAME DIRECTION - BOTH GOING STRAIGHT-SIDESWIPE	MOTOR VEHICLE IN TRANSPORT	OTHER (EXPLAIN IN NARRATIVE)
1/23/2010	1:47 pm	No	2	No	2	SAME DIRECTION - BOTH GOING STRAIGHT-REAR END	MOTOR VEHICLE IN TRANSPORT	FOLLOWED TOO CLOSELY
12/10/2010	6:41 am	No	1	No	0	ONE MOTOR VEHICLE - GOING STRAIGHT	OTHER OBJECT	OTHER (EXPLAIN IN NARRATIVE)
2/11/2010	9:50 am	No	1	No	0	ONE MOTOR VEHICLE - GOING STRAIGHT	FIXED OBJECT	FAILED TO CONTROL SPEED
2/11/2010	11:58 am	No	1	No	1	ONE MOTOR VEHICLE - GOING STRAIGHT	OVERTURNED	UNSAFE SPEED
2/21/2010	3:25 pm	No	2	No	0	SAME DIRECTION - BOTH GOING STRAIGHT-SIDESWIPE	MOTOR VEHICLE IN TRANSPORT	DRIVER INATTENTION
2/22/2010	6:20 pm	No	2	No	0	SAME DIRECTION - ONE STRAIGHT-ONE STOPPED	MOTOR VEHICLE IN TRANSPORT	FOLLOWED TOO CLOSELY
2/6/2010	11:51 am	No	2	No	0	OPPOSITE DIRECTION - ONE STRAIGHT-ONE LEFT TURN	MOTOR VEHICLE IN TRANSPORT	FAILED TO YIELD RIGHT OF WAY - TURNING LEFT
2/8/2010	8:10 am	No	1	No	0	ONE MOTOR VEHICLE - GOING STRAIGHT	FIXED OBJECT	
3/12/2010	3:50 pm	No	2	No	0	SAME DIRECTION - BOTH GOING STRAIGHT-SIDESWIPE	MOTOR VEHICLE IN TRANSPORT	FAILED TO DRIVE IN SINGLE LANE
3/20/2010	7:50 am	No	2	No	1	SAME DIRECTION - BOTH GOING STRAIGHT-SIDESWIPE	MOTOR VEHICLE IN TRANSPORT	
3/20/2010	9:10 am	No	1	No	0	ONE MOTOR VEHICLE - GOING STRAIGHT	FIXED OBJECT	
3/20/2010	7:29 am	No	1	No	0	ONE MOTOR VEHICLE - GOING STRAIGHT	FIXED OBJECT	
3/20/2010	6:15 am	No	2	No	0	SAME DIRECTION - BOTH GOING STRAIGHT-SIDESWIPE	MOTOR VEHICLE IN TRANSPORT	FAULTY EVASIVE ACTION
3/20/2010	6:38 am	No	1	No	1	ONE MOTOR VEHICLE - GOING STRAIGHT	FIXED OBJECT	
3/7/2010	6:17 pm	No	2	No	0	SAME DIRECTION - ONE STRAIGHT-ONE STOPPED	MOTOR VEHICLE IN TRANSPORT	DRIVER INATTENTION

This table shows a row for each crash record. The columns are the fields you selected in the field list editor. Note that you can always go back and change the selected fields.

See also: [Crash listings](#)¹⁰³, [Questions about lists](#)⁴⁷

Sorting the list

To sort a list, first **select the List Settings tab**.


The list may have up to three sort fields. If the field you are sorting on is numeric or date, be sure to specify that type in the combo box to the right of the field name. You can also select "Ascending" or "Descending" using the checkbox.

Select the "List report" tab to return to the newly sorted list.


2.9 Copy and Paste

Now that you've created diagrams, charts, and listings, and configured them to your liking it would be nice to be able to reuse these reports. There are a few ways you can do this, but one of the easiest, is to use the copy/paste functionality.

To start, **select the diagram** that you created a few moments ago. We left it with some extra wide spacing, and the legend turned off. Let's take this diagram, and use it with other data.

Click on copy  on the bottom toolbar. This will copy the current diagram to the clipboard.


Next, **create a new study** by clicking on the create study button on the top toolbar. Fill in the information on the settings tab. Be sure to select a different location than you did for your first study. You should now have a study that is ready for reports.

Click on the paste  button. This will paste a copy of the original diagram from the clipboard. Notice that it has new crashes in it (provided by the study), but keeps the settings from the original diagram (spacing between crashes and no legend).

Another great way to use copy and paste is at the study level. You can copy a study and all reports that fall under the study. This allows you to make several reports that you like, then copy/paste a study and simply change the study settings. Crash Magic uses the data provided by the studies, all reports under that study will be repopulated, but keep existing settings.

2.10 Save and load templates



Templates contain all the settings of an item such as a report. These settings can then be loaded anytime they are desired. For example, one might create a day of week chart. Then, that chart might be ordered by day, and then the colors set specifically. (i.e. Blue for Mondays, green for Tuesdays, etc.)

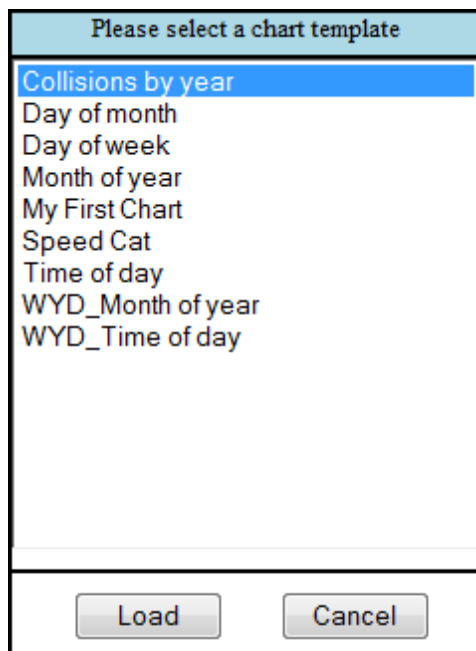
Earlier in this tutorial we created a chart. To get started let's select that chart. **Locate the chart in the project tree and click on it.** Let's create a template for this chart so that we can use it again in the future. To do this **click on the save template  button** on the bottom of the screen. You will now have the save template dialog box on the screen.

Please choose a name for this chart template

Existing templates:	Enter a new template name:
<ul style="list-style-type: none">Collisions by yearDay of monthDay of weekMonth of yearSpeed CatTime of dayWYD_Month of yearWYD_Time of day	<input type="text"/>
<input type="button" value="Replace existing template"/>	<input type="button" value="Create a new template"/>
<input type="button" value="Delete selected template"/>	

Under existing templates you will see a few chart templates provided by Pd' Programming, as well as any prepared for your group. To save this chart as a new template, **type "My First Chart"** under "Enter a new template name" and **click the "Create a new template" button**.

Next, we are going to create a new chart. **Click the new chart button**  at the top of the project tree. Just like the first time you created a chart, Crash Magic has made a default chart for you. We can now apply the template you saved to this new chart. **Click on the load template button**  at the bottom of the screen to bring up the list of existing chart templates.



You will see once again that this box is populated with several of the charts provided with Crash Magic, but you will also see a new entry "My First Chart". **Select "My First Chart" and click "Load"**. You should now have a chart that looks identical to the saved chart. This template will work for any chart in any study, so you can spend some time creating a very nice looking chart and reuse it many times.

This chart template is just the tip of the ice burg. Templates can be applied to studies, reports, filters, lists, etc.

See also: [Templates](#)  143

2.11 Use the filter


Now that we have generated a number of reports, it is time to take a look at limiting data. Crash Magic contains a powerful filtering tool that can limit data based on any fields in your database. These filters can be very simple (show only accidents with snow) to more complex (show only accidents with snow on Tuesdays from 10-11pm). Filters do not delete records from the database. They only limit the number of collisions displayed.

There are a few basic concepts that you need to understand about filters before we can begin:

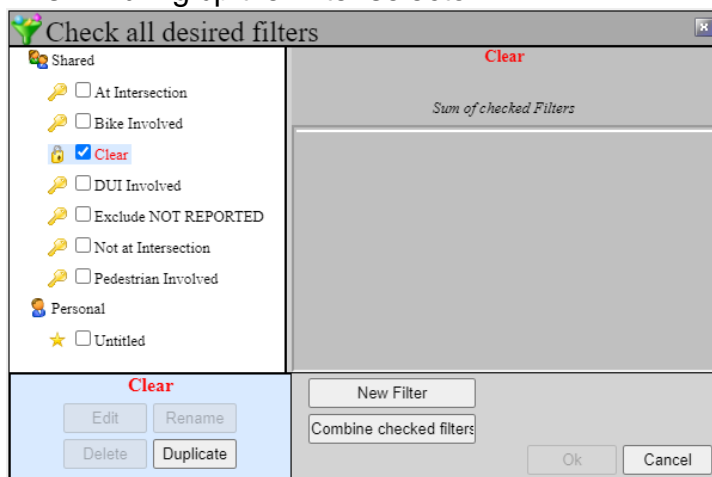
Filters will only reduce the number of records

It is important to note that filters only remove data from the study. A study is still limited by the date range and location on the settings tab, a filter is simply limiting that data.

Filters apply to all reports in a study

Filters are accessible for all reports and studies using the  button. Changing the filter will change the data not only in the report that you clicked the filter button on, but all reports that fall under the study in use.

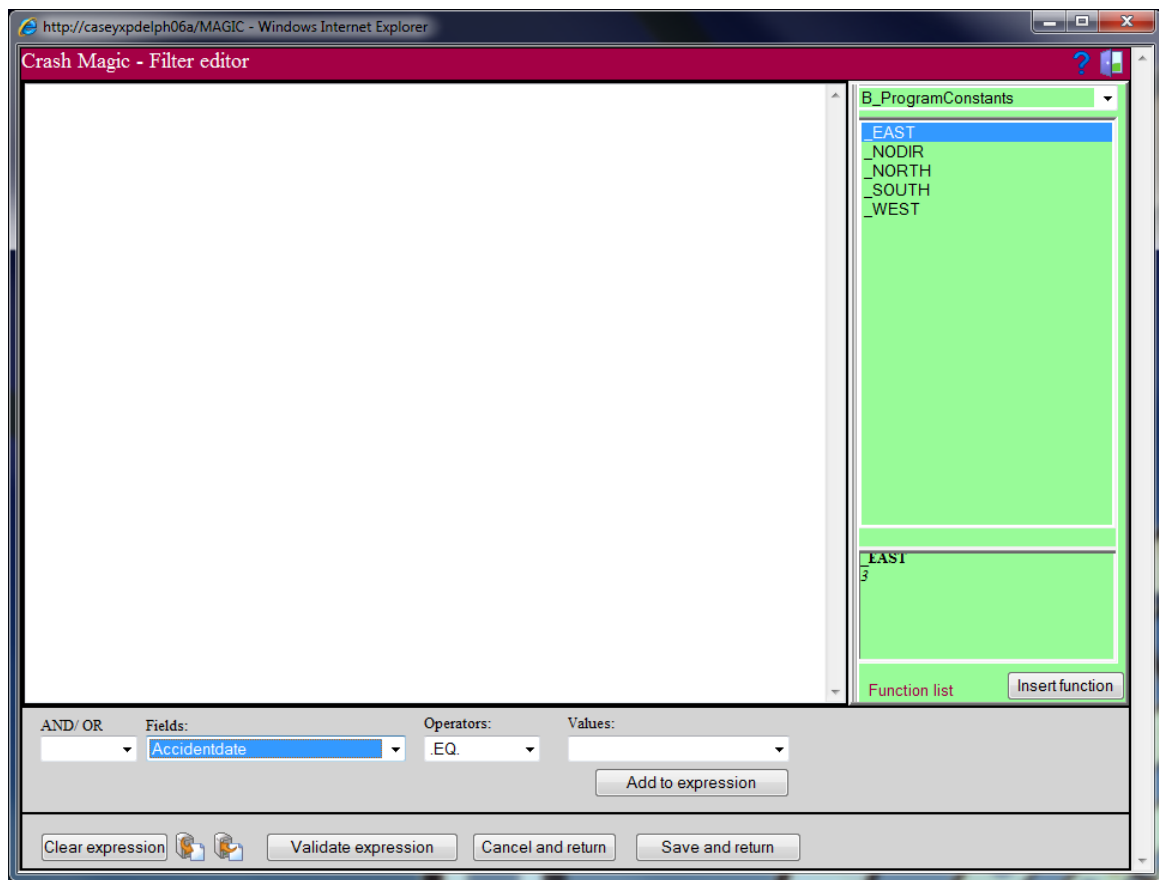
Time to write your first filter! To start **click the  button** at the bottom of any report. This will bring up the Filter selector.



The filter template selector allows users to store and reuse filters.

Click the New button to create a new filter. This will add an 'Untitled' filter to the listing. **Click the new 'Untitled' to activate it, then click the Edit button.**

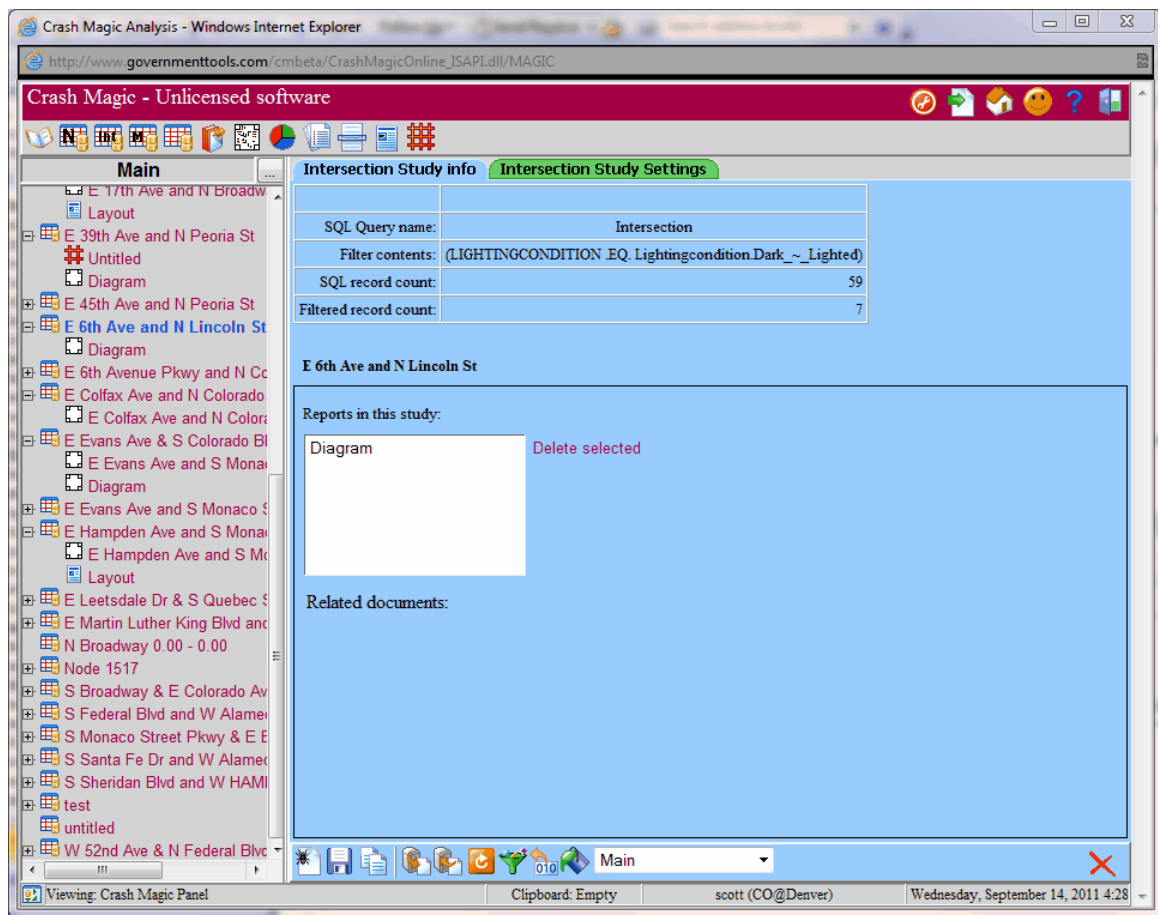
The edit filter window has three colored sections. The white area is where your actual filter will be displayed. The green section is for selecting advanced filter [functions](#)⁸⁰. The gray section contains the items used most frequently edit filters.




In this example we will write a simple filter. Filter fields and values are specific to the database Crash Magic is connected to. Your fields and values may be different. In this example we will write a filter for collisions that occurred when it was dark.

1. Click on the "Fields" drop down menu to view a list of fields from your database
2. Select the field in your database that contains the lighting condition (This could be light, lightcondition or other name for lighting condition in your database)
3. Leave the Operators field at ".EQ."
4. Click on the "Values" drop down menu
5. Select the value for a dark lighting condition(This could be dark, dark - lighted, or other value for a dark lighting condition)
6. Click the "Add to expression" button to insert the filter in the editor above
7. Click the "Save and return" button to see the results of the filter

The report or study being displayed will show fewer collisions with the filter applied.




In this example for a collision to be included in the study the field **LIGHTCONDITION** from the client collision database must have a value of **LightingconditionDark ~ Lighted** for the collision record to be included.


To remove a filter click on the filter button  to return to the filter selection. Click on the "Clear" filter. Then click the OK button to save the information.

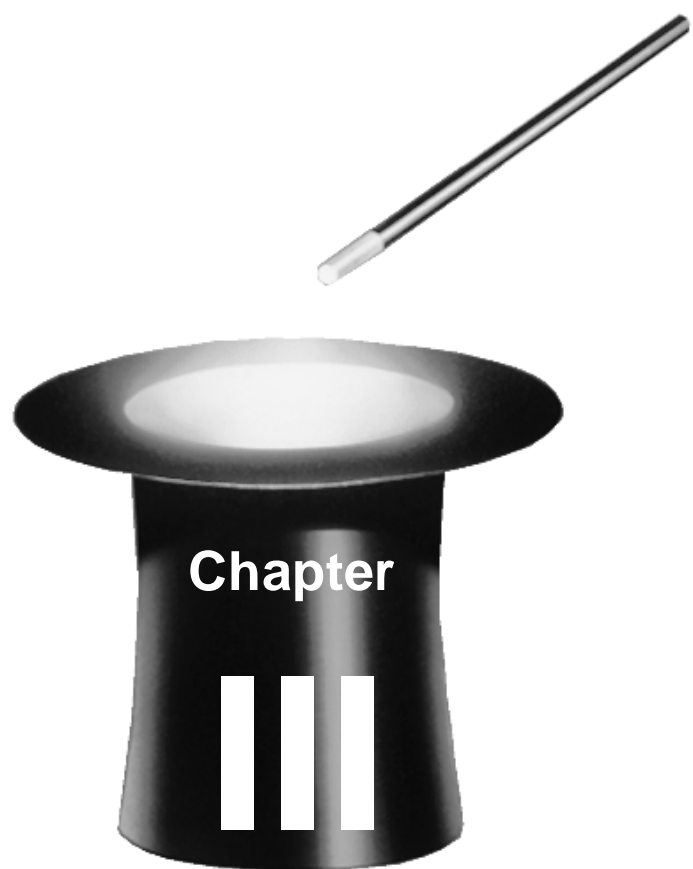
Filter conditions resolve to true or false. The filter is applied to each collision record in the study. If the result of applying the filter to the collision record is equal to true the record is included in the study. If the result is false, the record is not included in the study. Multiple conditions can be joined together using the "AND/OR" drop down menu.

2.12 Printing

All printing is done through PDFs. When the print button  is clicked Crash Magic sends a PDF to the PDF plug-in for printing. This will cause the browser to open the print dialog window for the PDF file. Users can select their printing preferences from this dialog window.

2.13 Log out

To log out of Crash Magic click on the Exit button  near the upper right corner of the page. Logging out of the program will save the page that you have been working on. You will need to enter your user id and password to enter back into Crash Magic.



3 How do I

3.1 Questions about logging in

Logging in should be as straight-forward as entering your user name and password. However, Crash Magic supports a number of methods of logging in, and there are several options one can use.

How do I create a shortcut or favorite to Crash Magic?

Crash Magic is a browser based application and is accessed by entering a URL into your browser. However, you will find that if you attempt to create a shortcut from the login form, you get an error when accessing that shortcut later. The trick is to remove the word "MAGIC" from the end of the URL. The "MAGIC" command is not part of the URL used to access the program.

There is a link in the lower right corner of the login form that will automatically create a "favorite" for you in your Internet Explorer browser. This link knows to remove "MAGIC" from the URL properly.

How do I start the Crash Magic program?

The Crash Magic software is actually always running. The software runs on a web server. That server may be within your agency, or it may be "hosted" on a server outside your company. In order to use the program, you must connect to it using a web browser.

With your web browser open, you must connect to the web site that hosts Crash Magic. If your agency has their own home page and links to applications, this can be one place to find a link to the program. If not, you will need to request a URL from the Crash Magic [group administrator](#)^[4] or project manager for the software. Once you enter the proper URL or click the link provided by your agency, you will be presented with the login form.

For information about the various options on the login form, see [Login form](#)^[87].

Is my login or password case sensitive?

Your login is not case sensitive, but your password is. Your password must be entered exactly as it was created.

My login or password isn't working

Crash Magic requires a login and password to access the program. The login is used by the program to keep track of your projects, studies and reports. The login process also assures that only one person is accessing your account at a time. The password keeps the information provided by the program protected. Your login and password are associated with a user group. In order to log in, you must select the user group that you have been assigned to.

When Crash Magic has been set up in a standard manner, the logins and passwords are stored in Crash Magic and managed by the [group administrator](#)^[4]. In most cases, if your password is not working, that is the person to contact. They will be able to verify that you have an account on the system, and they can change your password to one you can use. *Note: No one, even the group administrator, can look up your existing password. This is for security reasons.*

There are other methods of logging into the program. One method uses your SQL database for login verification. In this case, you still have a login to Crash Magic, but the password that you use is the password from your database login. The login screen will inform you that your user group provides this functionality by indicating "DB Verify available" once you have selected your user group. If your system is configured in this manner, your department database administrator is the person to see about lost or not-working passwords.

Another method of logging in is through use of "Active Directory" or "Single Sign On". If your network administrator has configured the Active Directory feature in Crash Magic, then your initial login to your computer account or to your enterprise network through your browser will eliminate the need to login to Crash Magic. In this case, merely clicking the link or entering the Crash Magic URL will bypass the login screen and present you with the project home page.


For information about the various options on the login form, see [Login form](#)^[87].

3.2 Questions about projects

Projects are used to group studies into based on the analysis being performed. The creation and management of projects is entirely optional, as all users are provided a project called "Main" and all work can be done in that project.

See also: [Projects](#)^[90]


How do I create a new project?

Projects are managed by clicking on the projects icon  in the top right corner of the screen. From there you can switch among projects and create new ones.

See also: [Projects](#)^[90]

How do I delete an existing project?

To delete a project, select the project by clicking on its name at the top of the screen.

Then, with the [project home page](#)^[90] displayed, click the delete button  in the lower right corner of the screen.

See also: [Projects](#)^[90], [Toolbar buttons](#)^[66]

3.3 Questions about studies

This "How Do I" section is about studies.

How can I export my crash records

Studies allow you to export a list of the crash records from the study. Here is how:

1. Select or create a study.
2. Click on the Study Settings tab.
3. Click the button Export the current study located at the bottom of the window.
4. Select or create a list of the fields from each crash record to export.
5. Click the OK button.
6. Save the file to an appropriate location.

3.4 Questions about reporting

This "How Do I" section is about reports.

How can I create a report button

Crash Magic allows group administrators to create buttons from templates that have been saved. Before a button can be created the template must be moved to the ".shared" user for everyone to access. The following steps assume that you have saved a template under your user and would like to create a button that everyone in your user group can use for the template.

The first step is to promote the template to the .shared user.

1. Log into Crash Magic as a group administrator.
2. In the admin tree on the left of the page click on the + sign next to your login name located under the name of your user group.
3. Click on the + sign under your login name next to the type of report template that you have saved (Note that in listings, charts, and cross tab reports the field list list reports or category list for charts and cross tab reports must also be moved as the reports are relying on access to these items).
4. Click on the name of the name of the template to share with the group.
5. Click your name in the drop down menu at the bottom of the page and select the .shared user.

Now that the report template has been moved, the following steps can be used to create a new report button.

1. While still logged in as group administrator, click on the + sign next to the .shared user under your user group name in the admin tree on the right
2. Click on the + sign next to the .options-reportbuttons.
3. Click on the Default.
4. Select the template that you just moved to the .shared user from the list of Available templates.
5. With your template selected Click the AddSelected button.
6. The Up and Down buttons can be used to position the button within the list of buttons.

Log out of the group administrator, and log into Crash Magic analysis. Click on a new study to see the report button.

3.5 Questions about collision diagrams

This "How Do I" section is about collision diagrams.

How can I change the curb lines

Crash Magic allows users to change the curb lines by selecting a new schematic. Here is how:

1. While on the diagram Click on the Diagram Settings tab.
2. Click the drop down menu labeled schematic.
3. Select the name of the schematic that is most appropriate for the location.

3.6 Questions about lists

This "How Do I" section is about lists.

Questions about charts

This "How Do I" section is about charts.

How can I change the fields displayed

Enter topic text here.

3.7 Questions about filters

This "How Do I" section is about lists.

How can I share my filter

Crash Magic allows for items to be shared in a user group. Each user group contains a user called the ".shared" user. Items in the .shared user can be accessed by everyone in the group. Crash Magic group administrators can move items like filters to the .shared user. The following steps assume that you have created a new filter under your user and would like to share it with your user group:

1. Log into Crash Magic as a group administrator.
2. In the admin tree on the left of the page click on the + sign next to your login name located under the name of your user group.
3. Click on the + sign next to the filter.template under your user name to display a list of your filters.
4. Click on the name of the filter that you would like to share with the group.
5. Click your name in the drop down menu at the bottom of the page and select the .shared user.


These steps will move the filter from your user to the .shared user. Other users will now be able to use the filter also.

3.8 Questions about users

This "How Do I" section is about users.

How do I add a user

Crash Magic allows for group administrators to add more users. Here is how:

1. Log into Crash Magic as a group administrator.
2. Click on the name of the group you have logged into in the project on the left of the page.
3. Click on the new user button .
4. Change the name field to the name of the user.
5. Change the Login field to the name the user will use to login(In cases where each user has their own login to the collision database, the Crash Magic login name must match the database login).
6. Enter a password for the new user(This should be left blank in cases where each user in the group has their own login to the collision database).
7. Check any of the User permissions boxes required by the user(Group Administrator should only be checked for those users that will need access to the Group Admin side of Crash Magic).
8. Click the Exit button in the upper right corner of the window to leave the group administrator window and save the user.

Provide the new user with the login, password and address for Crash Magic. The user will also need to be added to the cmUsers group if your IT department uses active directory. If the database is used to control access the user will require a login and password for the database.



4 Reference

4.1 Add in utilities

cm Map Magic for ArcGIS

cm Local Viewer

Enter topic text here.

cm Node Mapper

Enter topic text here.

4.2 Aliases

Crash Magic uses aliases to help correct and clean up data without changing your database. Currently, aliases can be used in two ways. The first is a [street name alias](#)^[50]. These aliases are used to point from a street name to another street name. The second is an [intersection alias](#)^[59]. These aliases are used to point from an intersection to another intersection.

Street name aliases

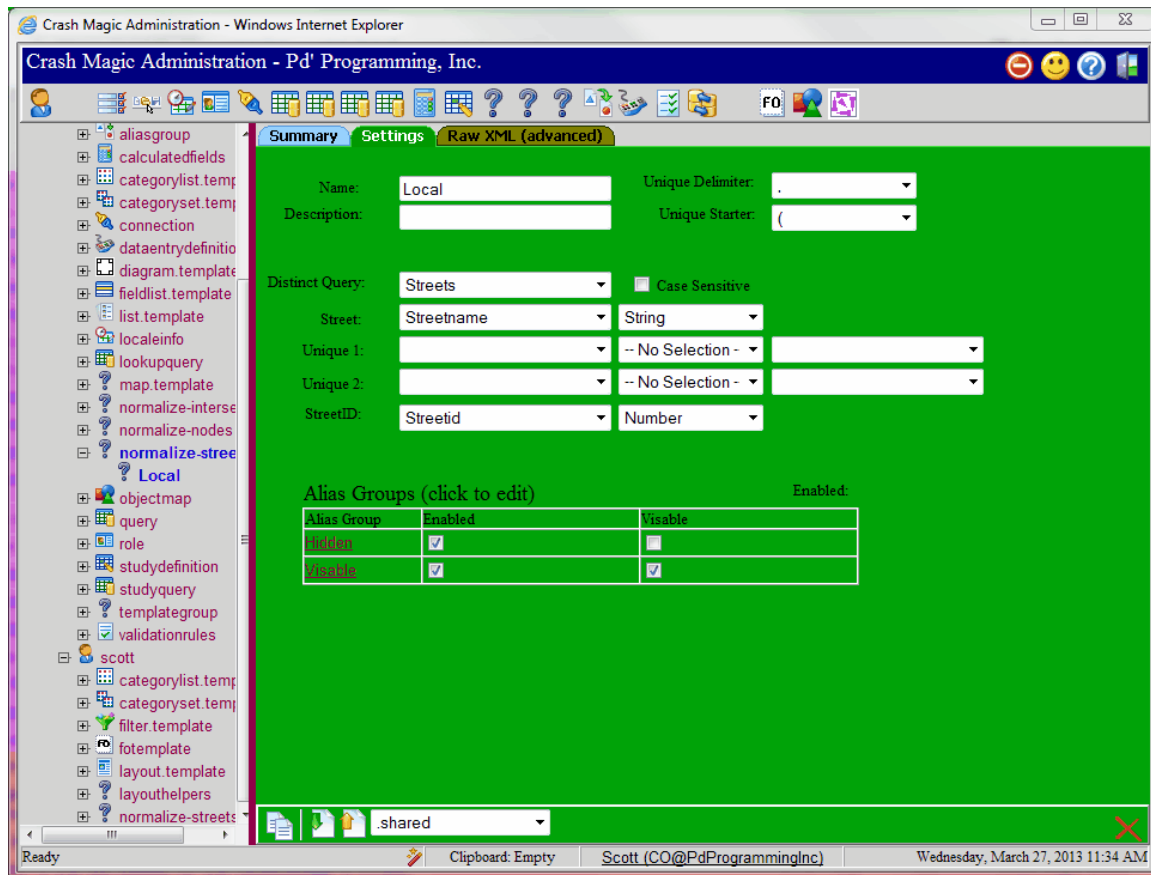
Street aliases are a powerful tool when used properly. This section will help you determine when it is a good time to use a street alias. A street alias allows two streets to be referenced as one street.

Street alias are an "always is" relationship. For example, "Badway" always is "Broadway". It is important that you only use street aliases when the location always is the other location. If you have a street (like a state highway) that might jump from one street to another, a street alias has the possibility of combining two intersections.

Street aliases can be more than one level deep. Allowing for you to alias Badway to Broadway, and then alias Broadway to Broadway Blvd.

Street aliases can only be created by Group Administrators. Alias group information is stored in an alias group object, but Street aliases are created and edited in the [streets normalizer](#)^[266]. A street alias will link two streets together so that requesting one street will gather the collisions from the other. The streets normalizers can be found in the group admin section of Crash Magic under the user group .shared user.

Most users will only have one street normalizer, but depending on the collision data in use multiple street normalizers can be found. Clients should review the study definition to determine the specific normalizer to be used.



In this example the Alias groups Hidden and Visible are enabled in the Local street normalizer.

Alias groups that are checked enabled will be used by the normalizer.

The visible check box determines whether the aliases should be displayed in the selection list. In the previous example a misspelled street name like Badway is not an actual street, and should not be in the list of selectable streets. Defining the alias in an alias group where Visible is unchecked will ensure that any streets selected as "From" in the alias group will not be displayed in the street selection list.

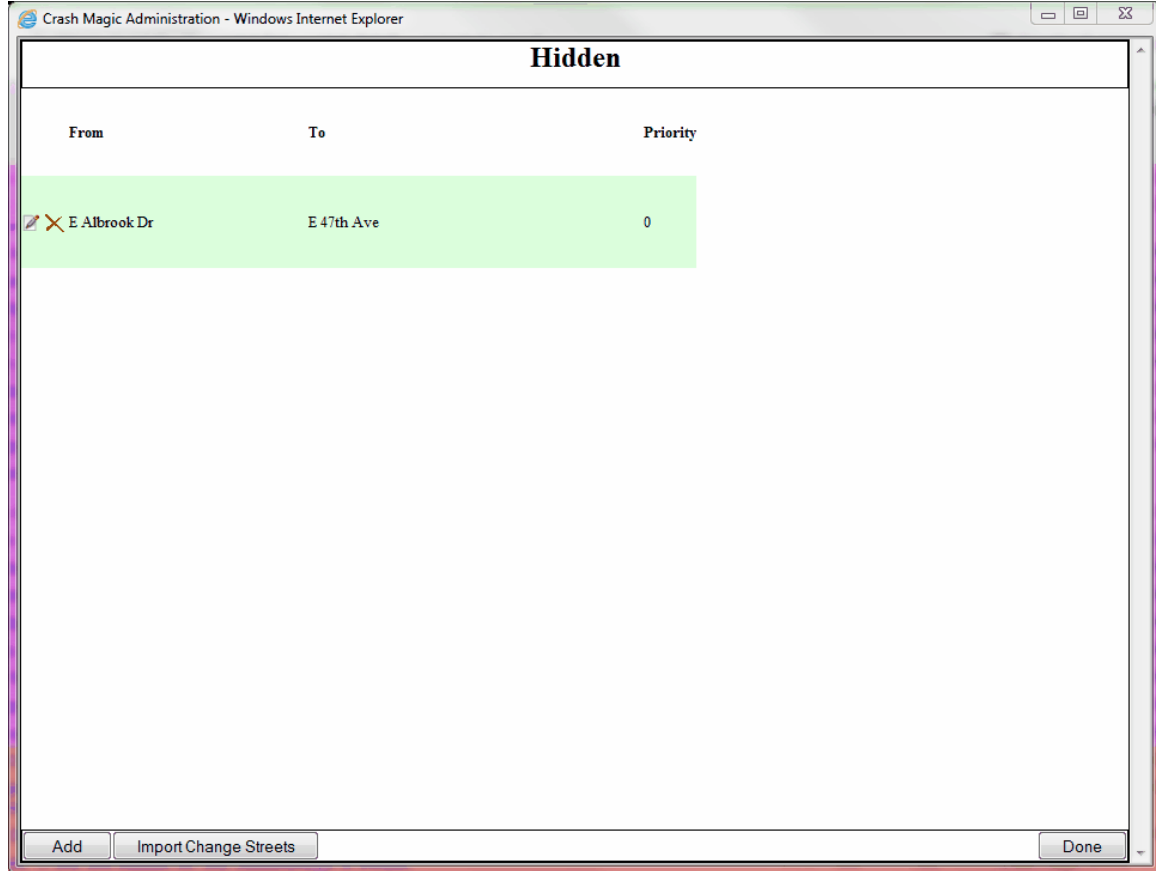
A street that has two valid street names should be placed in an alias group that is visible. For example if Hwy 85 is Broadway, and it is valid for a user to select either street name then an alias entry should be placed in a visible alias group. This will allow the user to see Hwy 85 and/or Broadway in a street selection box.

To add, delete or edit a street alias click on the alias group name to be taken to the [street alias](#) editor.

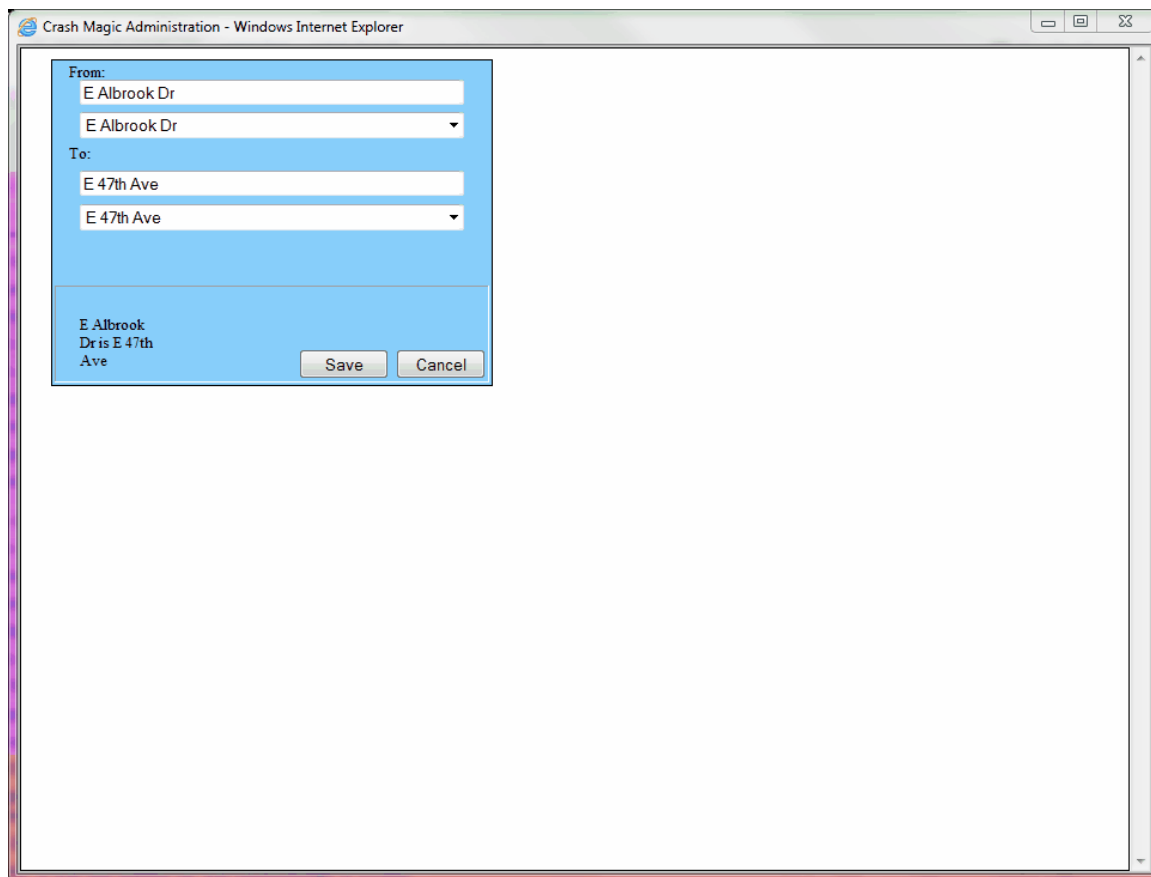
Edit form

Creating an alias defines a synonym between the two streets. If an alias is defined between A street and B street then using A street in a study includes B street and using B street in a study includes A street.

The street alias editor allows you to review, edit, and add new aliases.



Clicking the Add button will open the window to add a new street alias. Start typing a street name in the From or To box to see a list of selectable streets in the drop down list below. From streets defined in an alias group that is not visible will not be displayed in the street selection list of a study.



Crash Magic Administration - Windows Internet Explorer

From:
E Albrook Dr
E Albrook Dr

To:
E 47th Ave
E 47th Ave

E Albrook
Dr is E 47th
Ave

Save Cancel

In this example E Albrook Dr is being aliased to E 47th Ave. If the alias group is visible then using E Albrook Dr in a study will include E 47th Ave and using E 47th Ave in study will include E Albrook Dr. If the alias group is not visible then in this example E Albrook Dr would not be displayed as an available street when entering a street name into a study field.

Step by Step Aliases

Aliases in Crash Magic are the new way of taking care of misspelled street names, as well as tying different street names for one street together so that those crashes show up in a single query. Street aliases allow a street name to be associated with another name. This can be used in cases where a street changes names through an intersection, a street has multiple names, or even in cases where a bad name has been used for a street. For example, if a street named Broadway is also referred to as Hwy 36. An alias can be used to link the two streets. This means that when an alias is enabled, a study of an intersection of Broadway & Main St would also retrieve the crash records for Hwy 36 & Main St.

It is important to note that High Crash Location lists use favorite names to display the list of intersections. Using the previous example of Broadway aliased to Hwy 36.

Adding a street alias requires the group administrator privilege.

You must have access to the Administration side of Crash Magic in order to perform these steps.

1. Login to the Admin side of Crash Magic by typing in your User id and password and clicking on the Group Admin button (if this button isn't shown, click on the "show advanced options" in the lower right corner).

Crash Magic Login:

Userid:

Password: [Change...](#)

User group:



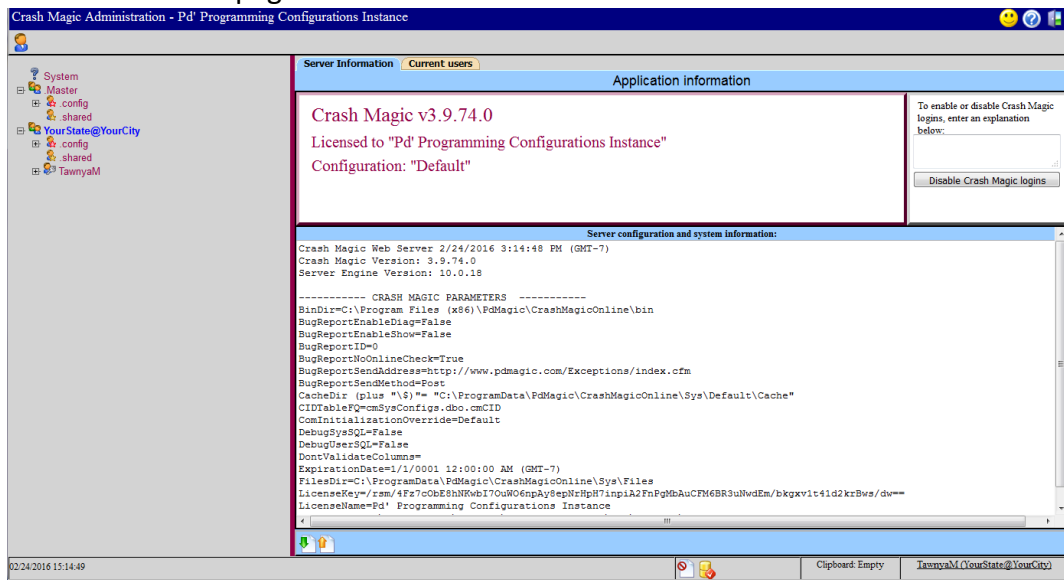
[Group Admin](#) [Master Admin](#)

[Maintenance Mode](#)
[Create user group](#)

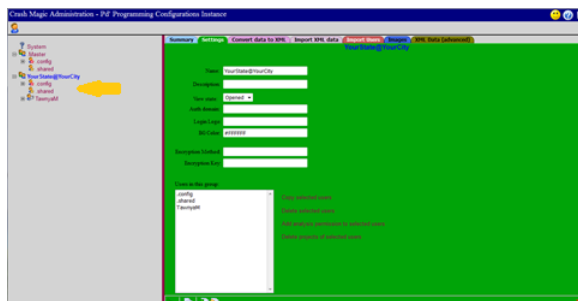
[Hide advanced options](#)



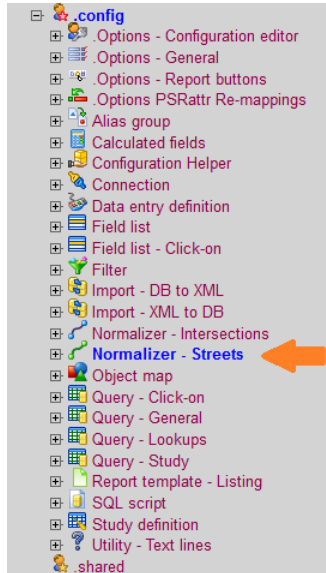
2. You should see a page similar to this:



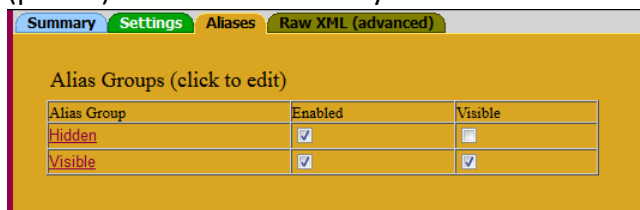
3. In order to get to the Aliases listing, click on the plus sign for .config under the city name from the tree on the left side.



4. Then click on the plus sign for Streets Normalizer:



5. Then click on the “Default” under the Normalizer Streets option, and you will see this (partial) screen. If necessary choose the Aliases tab at the top.



6. Click on the “Hidden” alias group, and you will see the listing of aliases being applied.

	W 32ND AVE {1}	W 32ND AVE		0
	W 45TH PLACE	W 45TH PL		0
	W 48TH AVE N	W 48TH AVE		0
	W 48TH AVE S	W 48TH AVE		0
	W 51ST PLACE	W 51ST PL		0
	W 38TH AVE.	W 38TH AVE		0
	W 26TH AVE.	W 26TH AVE		0
	W 26TH PL.	W 26TH PL		0
	W 27TH AVE.	W 27TH AVE		0
	W 28TH AVE.	W 28TH AVE		0
	W 28TH PL.	W 28TH PL		0
	W 29TH AVE.	W 29TH AVE		0
	W 29TH PL.	W 29TH PL		0
	W 30TH AVE.	W 30TH AVE		0
	W 30TH PL.	W 30TH PL		0
	W 31ST AVE.	W 31ST AVE		0
	W 31ST PL.	W 31ST PL		0
	W 32ND AVE.	W 32ND AVE		0
	W 32ND DR.	W 32ND DR		0
	W 32ND PL	W 32ND PL		0
Add		Import Change Streets		

7. To add a new alias (or misspelled street), click on the add button.

8. In the "From" option, type the misspelled street name, if it is in the database it will appear in the list below. Then in the "To:" option select what you want the street to appear as in the database. Then Save it.
- To delete an alias from the list, just click the red option next to the listing you want to delete at the aliases listing.
 - If you have a next to a listing, that's an indicator that the "From" street is not in the database. This is not necessarily an error, and is harmless to leave in the system. If that From street is ever added to the database, this alias will take effect automatically.
9. When finished, click on the button on the lower right side.

Alias reports

In some systems, the process of maintaining aliases is quite straight-forward. Perhaps a few streets have multiple names and a couple of streets a misspelling or two. In other systems, there could be thousands of misspelled street names. Street aliases are designed to handle both cases.

In order to make alias management as convenient as possible, there are a number of reports that can show your streets, aliases and combinations of both. These reports are accessed from the "Dependent data" tab of the street normalizer.

The terminology we use for aliases is shown in the graphic below.

- An "alias" is another name for a street. Just like a synonym. This is sometimes referred to as a "Bad Name".
- A "root" is a street name that has at least one alias pointing to it. This is often referred to as the "Good Name". There is no special setting to make a street into a root street - it simply becomes one by aliasing another name to it.
- A "standalone" name is a name that is neither an alias nor a root. In a typical database, most streets are standalone.

Summary

Settings

Dependent data

Raw XML (advanced)

CO@Boulder / .config (.config)
Normalizer - Streets (normalize-streets) / Default

Edit alias groups

Hidden

Visible

Export normalizer information

All street names

Aliases only

Root names only

Standalone names only

Roots and standalone names

Multi-line aliases

Single line aliases

BADWAY BLVD

Alias (also called "bad name")

BROADWAY

Root (also called "good name")

CENTRAL RD

Standalone

Verification reports

Please provide (upload) a "good streets" list:

Browse...

Compare data with good streets list

Alias Report types

<table><tr><th>A</th><th>B</th></tr><tr><td>10TH</td><td></td></tr><tr><td>10TH ST</td><td></td></tr><tr><td>1101 WALNUT ST</td><td></td></tr><tr><td>11ST</td><td></td></tr><tr><td>11TH</td><td></td></tr><tr><td>11TH ST</td><td></td></tr><tr><td>12TH ST</td><td></td></tr><tr><td>13TH</td><td></td></tr><tr><td>13TH ST</td><td></td></tr><tr><td>14TH</td><td></td></tr><tr><td>14TH ST</td><td></td></tr><tr><td>15TH</td><td></td></tr><tr><td>15TH ST</td><td></td></tr><tr><td>16TH</td><td></td></tr><tr><td>16TH CIR</td><td></td></tr><tr><td>16TH ST</td><td></td></tr><tr><td>1700-BL BLUFF ST</td><td></td></tr><tr><td>17TH</td><td></td></tr><tr><td>17TH ST</td><td></td></tr></table> <div>All street names</div>	A	B	10TH		10TH ST		1101 WALNUT ST		11ST		11TH		11TH ST		12TH ST		13TH		13TH ST		14TH		14TH ST		15TH		15TH ST		16TH		16TH CIR		16TH ST		1700-BL BLUFF ST		17TH		17TH ST		<table><tr><th>A</th><th>B</th></tr><tr><td>10TH</td><td></td></tr><tr><td>11ST</td><td></td></tr><tr><td>11TH</td><td></td></tr><tr><td>13TH</td><td></td></tr><tr><td>14TH</td><td></td></tr><tr><td>15TH</td><td></td></tr><tr><td>16TH</td><td></td></tr><tr><td>17TH</td><td></td></tr><tr><td>19TH</td><td></td></tr><tr><td>20TH</td><td></td></tr><tr><td>21ST</td><td></td></tr><tr><td>22ND</td><td></td></tr><tr><td>23RD</td><td></td></tr><tr><td>23RDT ST</td><td></td></tr><tr><td>24TH</td><td></td></tr><tr><td>24TH PL</td><td></td></tr><tr><td>26TH</td><td></td></tr><tr><td>27TH</td><td></td></tr><tr><td>28TH</td><td></td></tr><tr><td>28TH FRONTAGE RD</td><td></td></tr></table> <div>Aliases only</div>	A	B	10TH		11ST		11TH		13TH		14TH		15TH		16TH		17TH		19TH		20TH		21ST		22ND		23RD		23RDT ST		24TH		24TH PL		26TH		27TH		28TH		28TH FRONTAGE RD		<table><tr><th>A</th><th>B</th></tr><tr><td>10TH ST</td><td></td></tr><tr><td>11TH ST</td><td></td></tr><tr><td>13TH ST</td><td></td></tr><tr><td>14TH ST</td><td></td></tr><tr><td>15TH ST</td><td></td></tr><tr><td>16TH ST</td><td></td></tr><tr><td>17TH ST</td><td></td></tr><tr><td>19TH ST</td><td></td></tr><tr><td>20TH ST</td><td></td></tr><tr><td>21ST ST</td><td></td></tr><tr><td>22ND ST</td><td></td></tr><tr><td>23RD ST</td><td></td></tr><tr><td>24TH ST</td><td></td></tr><tr><td>26TH ST</td><td></td></tr><tr><td>27TH ST</td><td></td></tr><tr><td>28TH ST</td><td></td></tr><tr><td>28TH ST FRONTAGE RD</td><td></td></tr><tr><td>29TH ST</td><td></td></tr><tr><td>30TH ST</td><td></td></tr><tr><td>31ST ST</td><td></td></tr></table> <div>Root names only</div>	A	B	10TH ST		11TH ST		13TH ST		14TH ST		15TH ST		16TH ST		17TH ST		19TH ST		20TH ST		21ST ST		22ND ST		23RD ST		24TH ST		26TH ST		27TH ST		28TH ST		28TH ST FRONTAGE RD		29TH ST		30TH ST		31ST ST		<table><tr><th>A</th><th>B</th></tr><tr><td>1101 WALNUT ST</td><td></td></tr><tr><td>12TH ST</td><td></td></tr><tr><td>16TH CIR</td><td></td></tr><tr><td>1700-BL BLUFF ST</td><td></td></tr><tr><td>17TH ST</td><td></td></tr><tr><td>18TH</td><td></td></tr><tr><td>18TH ST</td><td></td></tr><tr><td>19TH ST</td><td></td></tr><tr><td>19TH ST</td><td></td></tr><tr><td>19TH ST</td><td></td></tr><tr><td>19TH ST</td><td></td></tr><tr><td>25TH ST</td><td></td></tr><tr><td>27H WY</td><td></td></tr><tr><td>27TH WAY</td><td></td></tr><tr><td>28TH FRONTAGE</td><td></td></tr><tr><td>28TH ST ON RAMP</td><td></td></tr><tr><td>28TH ST OVERPASS</td><td></td></tr><tr><td>28TH ST/HWY 36</td><td></td></tr></table> <div>Standalone names only</div>	A	B	1101 WALNUT ST		12TH ST		16TH CIR		1700-BL BLUFF ST		17TH ST		18TH		18TH ST		19TH ST		19TH ST		19TH ST		19TH ST		25TH ST		27H WY		27TH WAY		28TH FRONTAGE		28TH ST ON RAMP		28TH ST OVERPASS		28TH ST/HWY 36	
A	B																																																																																																																																																																				
10TH																																																																																																																																																																					
10TH ST																																																																																																																																																																					
1101 WALNUT ST																																																																																																																																																																					
11ST																																																																																																																																																																					
11TH																																																																																																																																																																					
11TH ST																																																																																																																																																																					
12TH ST																																																																																																																																																																					
13TH																																																																																																																																																																					
13TH ST																																																																																																																																																																					
14TH																																																																																																																																																																					
14TH ST																																																																																																																																																																					
15TH																																																																																																																																																																					
15TH ST																																																																																																																																																																					
16TH																																																																																																																																																																					
16TH CIR																																																																																																																																																																					
16TH ST																																																																																																																																																																					
1700-BL BLUFF ST																																																																																																																																																																					
17TH																																																																																																																																																																					
17TH ST																																																																																																																																																																					
A	B																																																																																																																																																																				
10TH																																																																																																																																																																					
11ST																																																																																																																																																																					
11TH																																																																																																																																																																					
13TH																																																																																																																																																																					
14TH																																																																																																																																																																					
15TH																																																																																																																																																																					
16TH																																																																																																																																																																					
17TH																																																																																																																																																																					
19TH																																																																																																																																																																					
20TH																																																																																																																																																																					
21ST																																																																																																																																																																					
22ND																																																																																																																																																																					
23RD																																																																																																																																																																					
23RDT ST																																																																																																																																																																					
24TH																																																																																																																																																																					
24TH PL																																																																																																																																																																					
26TH																																																																																																																																																																					
27TH																																																																																																																																																																					
28TH																																																																																																																																																																					
28TH FRONTAGE RD																																																																																																																																																																					
A	B																																																																																																																																																																				
10TH ST																																																																																																																																																																					
11TH ST																																																																																																																																																																					
13TH ST																																																																																																																																																																					
14TH ST																																																																																																																																																																					
15TH ST																																																																																																																																																																					
16TH ST																																																																																																																																																																					
17TH ST																																																																																																																																																																					
19TH ST																																																																																																																																																																					
20TH ST																																																																																																																																																																					
21ST ST																																																																																																																																																																					
22ND ST																																																																																																																																																																					
23RD ST																																																																																																																																																																					
24TH ST																																																																																																																																																																					
26TH ST																																																																																																																																																																					
27TH ST																																																																																																																																																																					
28TH ST																																																																																																																																																																					
28TH ST FRONTAGE RD																																																																																																																																																																					
29TH ST																																																																																																																																																																					
30TH ST																																																																																																																																																																					
31ST ST																																																																																																																																																																					
A	B																																																																																																																																																																				
1101 WALNUT ST																																																																																																																																																																					
12TH ST																																																																																																																																																																					
16TH CIR																																																																																																																																																																					
1700-BL BLUFF ST																																																																																																																																																																					
17TH ST																																																																																																																																																																					
18TH																																																																																																																																																																					
18TH ST																																																																																																																																																																					
19TH ST																																																																																																																																																																					
19TH ST																																																																																																																																																																					
19TH ST																																																																																																																																																																					
19TH ST																																																																																																																																																																					
25TH ST																																																																																																																																																																					
27H WY																																																																																																																																																																					
27TH WAY																																																																																																																																																																					
28TH FRONTAGE																																																																																																																																																																					
28TH ST ON RAMP																																																																																																																																																																					
28TH ST OVERPASS																																																																																																																																																																					
28TH ST/HWY 36																																																																																																																																																																					

<div> <div>A</div> <div> 1 2 1101 WALNUT ST 3 12TH ST 4 16TH CIR 5 1700-BL BLUFF ST 6 17TH ST 7 18TH 8 18TH ST 9 19TH ST 10 19TH ST 11 19TH ST 12 19TH ST 13 19TH ST 14 25TH ST 15 27H WY 16 27TH WAY 17 28TH FRONTAGE 18 28TH ST ON RAMP 19 28TH ST OVERPASS 20 28TH ST/HWY 36 </div> </div> <div>Root and standalone names</div>	<div> <div> 30 21ST ST 21 22 22ND ST 23 24 23RD ST 25 26 27 24TH ST 28 29 30 26TH ST 31 32 33 34 27TH ST 35 36 28TH ST 37 38 39 28TH ST FRONTAGE RD 40 41 42 </div> <div> 21ST 22ND 23RD 24TH 25RD 26RD 27TH 28TH 29TH 30TH 31TH 32TH 33TH 34TH 35TH 36TH 37TH 38TH 39TH 40TH 41TH 42TH </div> </div> <div>Multi-line aliases</div>	<div> <div> 10 21ST ST 11 22ND ST 12 23RD ST 13 24TH ST 14 26TH ST 15 27TH ST 16 28TH ST 17 28TH ST FRONTAGE RD 18 29TH ST 19 30TH ST 20 31ST ST 21 32ND ST 22 33RD ST </div> <div> 21ST 22ND 23RD 24TH 26TH 27TH 28TH 28TH FRONTAGE RD 29TH 30TH 31ST ST 32ND ST 33RD ST </div> <div> 23RD ST 24TH PL N 26TH ST N. 26TH 28TH. ST 28TH FRONTAGE RD 28TH ST FRONTAGE 28TH STREET FRONTAGE RD </div> </div> <div>Single-line aliases</div>
--	---	--

Verification reports

In some cases, there may be a pre-defined list of "good" street names. Perhaps this is from a GIS, or from another database that has been carefully cleaned and reflects all the streets within the jurisdiction of the agency. The verification report can be used to compare that pre-defined list with the current database and aliases. This will provide a report listing the names that must be cleaned.

The verification report is found on the same tab as the alias reports. That is, inside the Normalizer - Streets resource.

CO@Boulder / .config (.config)
Normalizer - Streets (normalize-streets) / Default

Edit alias groups

Hidden

Visible

Export normalizer information

All street names

Aliases only

Root names only

Standalone names only

Roots and standalone names

Multi-line aliases

Single line aliases

BADWAY BLVD
Alias (also called "bad name")

BROADWAY
Root (also called "good name")

CENTRAL RD
Standalone

Verification reports

Please provide (upload) a "good streets" list:

Browse...

Compare data with good streets list

Note: if such a pre-defined list does not exist, one can be created by using the other reports available on this page. One option is the "All street names" report and eliminate the "bad names". Another option is to start with the "Roots and standalone names" report.

To produce a verification report:

1. Create or obtain a list of all of the "good" streets for your jurisdiction. Put this in a text file, one name per line.
2. Click the "Browse" button to select your list of good streets.
3. Click the "Compare data with good streets list" to produce the report.

	A	B	C	D	E
1	StreetName	InGoodList	IsAlias	NeedsWork	
2		FALSE	FALSE	TRUE	
3	10TH	FALSE	TRUE	FALSE	
4	10TH ST	TRUE	FALSE	FALSE	
5	1101 WALNUT ST	TRUE	FALSE	FALSE	
6	11ST	FALSE	TRUE	FALSE	
7	11TH	FALSE	TRUE	FALSE	
8	11TH ST	TRUE	FALSE	FALSE	
9	12TH ST	TRUE	FALSE	FALSE	
10	13TH	FALSE	TRUE	FALSE	
11	13TH ST	TRUE	FALSE	FALSE	
12	14TH	FALSE	TRUE	FALSE	
13	14TH ST	TRUE	FALSE	FALSE	
14	15TH	FALSE	TRUE	FALSE	

Verification report output

The verification report contains 4 columns.

- StreetName - The names of all the streets in the database
- InGoodList - TRUE if StreetName is in the "good" streets file that was uploaded
- IsAlias - TRUE if StreetName is already defined as an alias
- NeedsWork - TRUE if StreetName is not in the "good" list, and not an alias. FALSE otherwise.

Two strategic options:

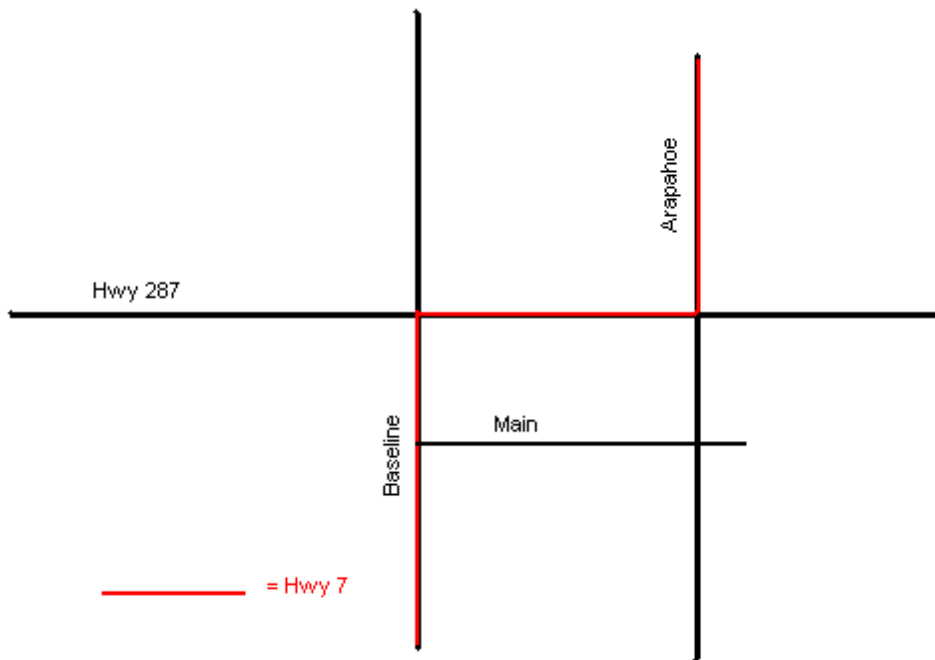
1. If aliases are being used to correct street names, then the NeedsWork column is the best indicator of streets that should be corrected.
2. If the intent is to correct directly, without aliases, all that is needed is the InGoodList column.

NOTE: In Crash Magic versions prior to 3.9.181, this report does not have headers and the TRUE/FALSE values are shown as 0/-1. To see the report as it is above, add the headers and highlight columns B,C,D. Then search and replace -1 to TRUE and 0 to FALSE.

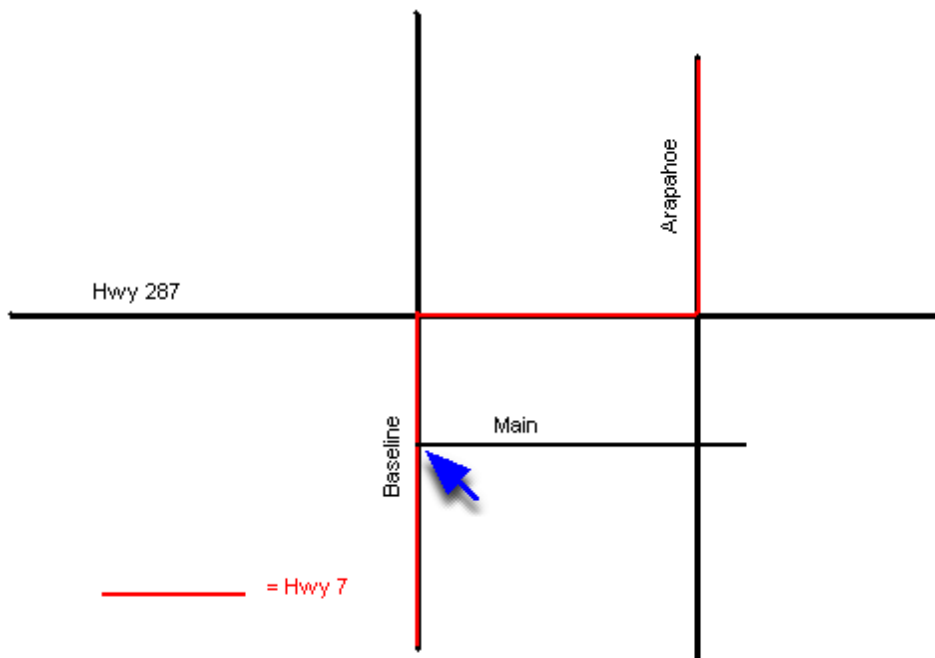
Intersection aliases

IMPORTANT - Intersection Aliases do not currently exist for end-users of the program.

Intersections aliases are more specific than street aliases. Instead of aliasing an entire street, you pick a single intersection to alias to another. For our example we are going to use the location below:



The red line is Hwy 7, which runs on a few roads, including Baseline and Arapahoe. If you were to alias Hwy 7 to either of these roads directly (using a street alias) when asking for Hwy 7 & Main you would get the crashes at both Baseline & Main, and Arapahoe & Main in one study. Instead of using a street alias for this location we will use an intersection alias. For this example we are trying to get all the crashes that occur at Hwy 7 & Main:



We will make create an alias that says Hwy 7 & Main = Baseline & Main. Now all crashes that are coded to Baseline & Main, and Hwy7 & Main will be present in the study. All crashes that occur at Arapahoe & Main will not.

Edit form

Main Edit Screen

Replace Category - The user defined category that identifies who will be using the aliases. Clicking on any of the categories will bring you to the Category Edit Screen.

Active - If checked, the current category of aliases is being used program wide.

Category Edit Screen

Purpose - User defined purpose that identifies what the alias is used for. Changing this box will display aliases for the selected purpose. Selecting "All" will show all aliases in the current category.

Delete - Deletes the rule.

Edit - Edits the rule.

Mousing over any street will display more information about that location.

Rule Add/Edit Screen

From - Defines the criteria that a street must match to be aliased.

To - Defines the root street.

Street Name - The name of the street.

Unique1/Unique2 - Unique values that are used to ensure the location is part of the correct municipality.

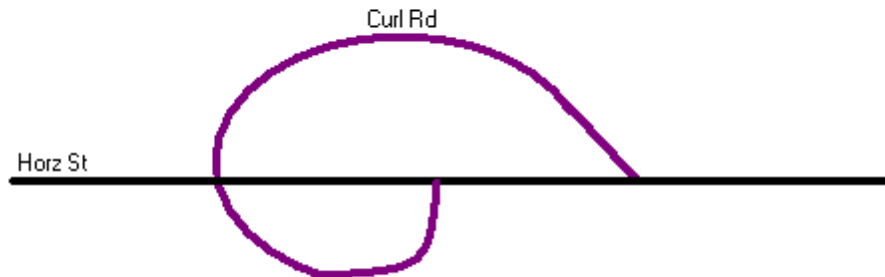
Purpose - User defined purpose that identifies what the alias is used for.

Node aliases

Node aliases, like street and intersection aliases are used to map multiple nodes to a single one where all the nodes represent the same location. This is a situation that should be rarely if ever used.



Duplicate intersections

Split locations are used when an intersection name has more than one possible physical location. The example below has three physical locations that all have the same intersection Curl Rd & Horz St.



Split locations allow you to flag an intersection as one that has more than one physical location, and allows you to manually split the crashes into separate locations.

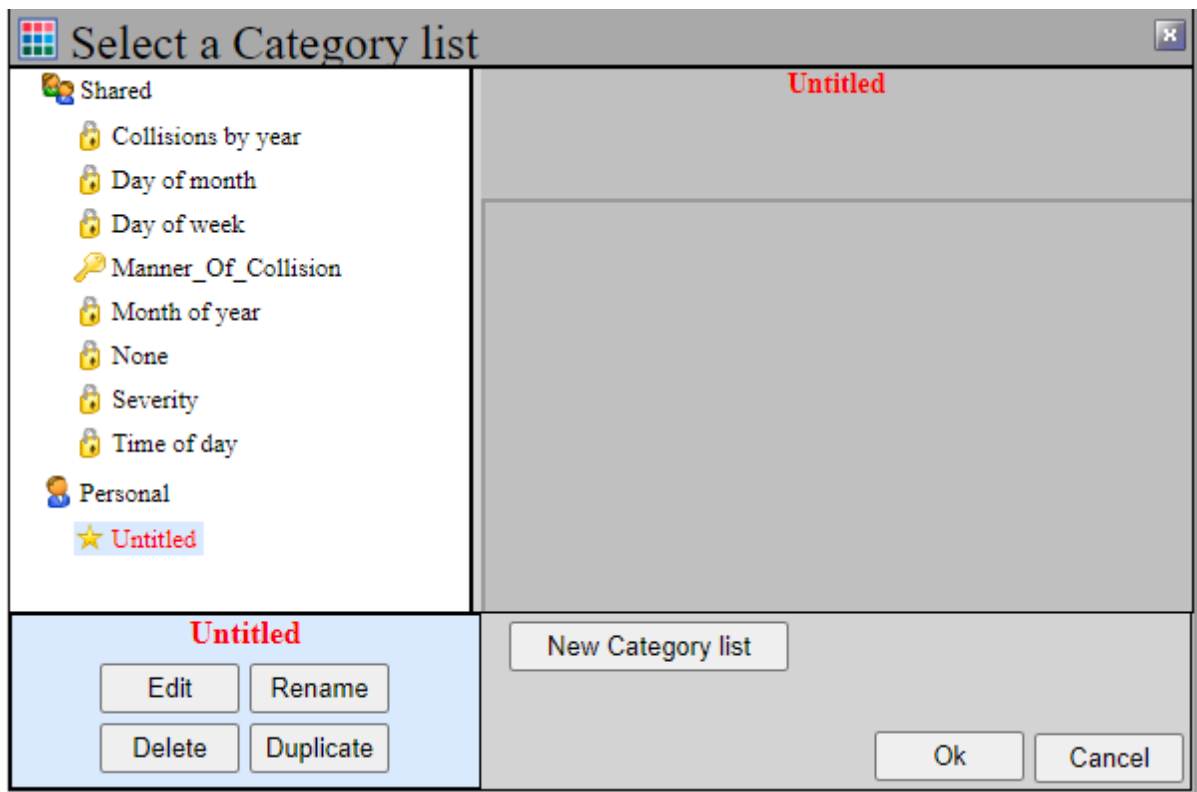
4.3 Category List

Crash Magic has a great display feature called Category lists. Category lists allows records to be highlighted based on the data within the record. Category lists are tied to the study and affect the diagram and listing reports. Category lists are also used in the display of chart information. Category lists can be selected and edited with the "Change change study category list" button  or the category list button  on the chart settings tab.

Category lists can be thought of as groups of bins. Each bin in a Category list is used for counting and coloring records. As Crash Magic pulls each record the record is compared against the conditions for each bin. If the condition resolves to true for a bin the record is counted in the bin and the record is colored with the bin color. If a record resolves to true for more than one bin in a single category list an error message will be displayed to the user when the category list is used.

Category list selector

The category list selector allows the users to select or create a category list. Each list is displayed under the owner tree.



New - Creates a new blank category list named Untitled

Del - Deletes the currently selected category list

Edit Category list - Opens the category list editor for the selected category list

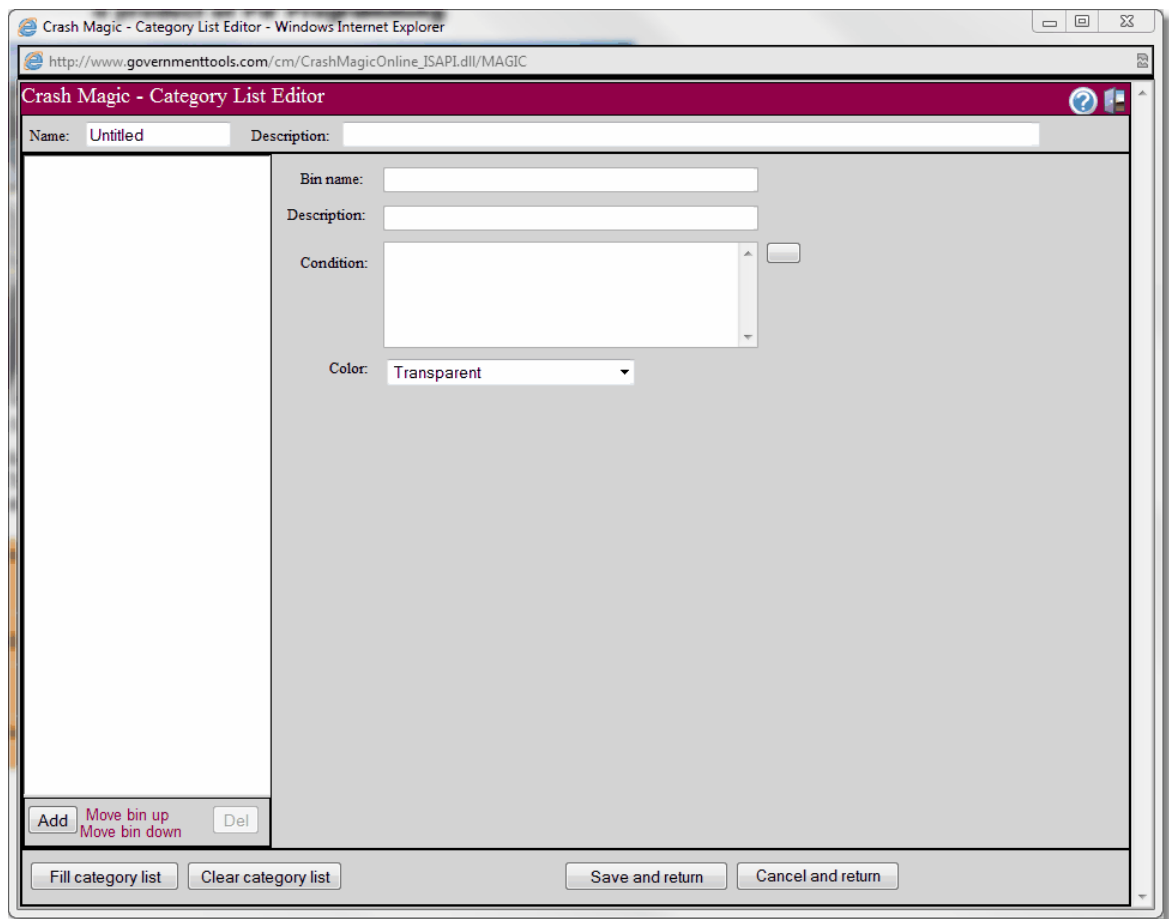
Ok - Closes the category selection list box and sets the category list to the current category list selected

Cancel - Closes the category selection list box without selecting the category list

Category list editor

The category list editor allows users to create and edit new category lists. Users can select fields in their database to build category lists from, change colors for each bin. The category list editor page is divided into four sections. The top section displays the name and description of the category list. The left section displays the list of bins for the current category list. The right section displays information on the currently selected bin. The bottom section is for overall functions of the category list.

Most clients will build category lists by clicking the Fill category list button and selecting a field from the database that Crash Magic will build the bin list from.



Name - Name of the category list being edited(The Name will default to Untitled)

Description - Description of the category list

Add - This button adds a new category list bin

Move bin up - This link will move the currently selected bin up in the order of bins

Move bin down - This link will move the currently selected bin down in the order of bins

Del - This button deletes the currently selected category list bin

Bin name - This field displays the name of the currently selected bin

Bin description - This field displays the a description of the currently selected bin

Condition - The condition in which a record is placed in the current bin(As with filters each condition must resolved to true or false for a single record)

Editor button - This button opens the filter editor for editing the condition of the bin

Color - This is the color selected for the current bin

Fill category list - This button opens a dialog to select a field from the database that will populate the list of bins

Clear category list - This button will clear the current list of bins

Save and return - Saves the current category list of bins and closes the editor

Cancel and return - Closes the editor without saving any changes made to the category list

4.4 Common interface elements

Logout, help and home buttons

In the top right are buttons to exit, get help, or open the main project page.



Exit - Exit Crash Magic analysis, and return to the login form.



Help - Open the help navigator.



Administration - Open the Crash Magic administration forms.

New report buttons

To generate a report, the user must first select a study. When a study has been selected, the report buttons become available at the top of the screen.

Note: The specific report buttons available are determined by the configuration and administration settings.

Report buttons



Apply template group - This button allows a set of predefined reports to be created for a study. This button may not be active for all users.



Create diagram - Creates a new diagram report with the default settings based on the collisions from the study.



Create chart - Create a new chart report based on the collisions from the study.



Create list - Displays user selected fields of each collision record based on the collisions from the study.



Create location list - Creates a list of high crash locations based on the locations from the study. This button is only active for users that locate their collisions by a location id or street and cross street.



Create layout - Creates a layout report. This allows users to combine different report elements into a single report.



Create cross tab report - Creates a cross tab report.

Common toolbar buttons

Below the display of each project, study and report in the program is a toolbar. The buttons on the toolbar change depending on the content being displayed at the time. The following is a list of common buttons used by Crash Magic with a description of their function.



Filter - This button raises the filter editor form. Each study has its own filter, which applies to all reports within that study. Thus, changing the filter will affect all reports in the current study.

The filter is used to remove crashes from the current study. It will never cause more crashes to be included than exist in a study without a filter.

For more information of filters, see the [Filters](#)^[76] chapter.



Category - This button opens the category list editor. [Category list](#)^[62] allow users to highlight diagram and listing reports for the current study.



Re-query study - Will cause Crash Magic to query the database again. Any changes made to the crash data will be displayed in the current study or report.



Print - Selecting this button causes Crash Magic to prepare an Adobe Acrobat® file and present it in a popup window.

Printing is available from the new popup window.

This same report is available in the preview tab of the current report. In this case, no popup window is created.

Note: If your computer has "popup-blocking" software installed, you may not see this new window. Most popup blocking software will allow such popups when the control key is held down during the click.



Copy - Pressing this button creates a copy of the current Project, Study or Report. The copy is kept in the Crash Magic clipboard until the user presses the Paste button. Note: once something is copied to the clipboard, it's name and description appear in the status bar at the bottom of the window.



Paste - Pressing this button pastes a copy of the item from the clipboard into the current branch of the project tree. In order to paste an item, an appropriate tree branch must be selected. For example, a report can not be pasted into a project: it may only be pasted into a study. However, a project may be pasted anywhere, and will appear at the highest level of the tree.



Load template - This button displays a dialog from which the user can select a template to load for the current project, study or report. A template is used to fill in saved values for a project, study or report. For example, a chart template contains the chart categories, colors, style, etc. This makes it easy to save and retrieve, for example, a horizontal bar chart showing time of day in shades of light to dark blue.

For more information, see the [Templates](#)^[143] section.



Save template - This button displays a dialog from which the user can create a new template, or overwrite an existing one.

For more information, see the [Templates](#)^[143] section.



Delete - Deletes the current project, study or report shown. Important: Items are not retrievable once they have been deleted.

4.5 Crash Magic Local Viewer

The Crash Magic Local Viewer (cmLocalViewer) is a desktop utility that can be used to display crash reports, photos, or other information as requested by Crash Magic. In most cases, scanned crash reports or photos are stored locally at the agency utilizing Crash Magic. For security and data size reasons, it is usually best to avoid uploading them to the hosted server. The cmLocalViewer makes access to these local files safe and efficient without transmitting personal/confidential information outside the local network.

The cmLocalViewer is installed as a desktop application, registered in Windows to handle files with the "cmlvml" extension. Crash Magic includes a feature to write such files containing information about the file/program to launch on the local computer. cmLocalViewer is currently capable of responding to two types of requests.

Because cmLocalViewer runs in the security context of the user currently logged into Windows, it is limited to presenting content that user has access to. It is limited by the same security provided by network devices and/or Active Directory.

Two new functions have been added to the Crash Magic expression parser to facilitate the creation of the cmlvml content. Rather than piecing together the XML manually, users may instead utilize the cmlvmlLocalFile and cmlvmlLocalExe functions. These two functions assemble the required XML for most uses of the program.

LocalFile

LocalFile is used to present the end user with a specific file or list of files located on their local computer, or network server.

This type of request is accompanied by a list of local file paths to search, and a list of "specs" or file masks. For example, a request might look like this:

```
<pdroot>
  <cmLocalViewer>
    <Command>LocalFile</Command>
    <LocalFile>
      <Paths>
        <Path>\\FileServer1\CrashData</Path>
        <Path>\\FileServer1\Supplemental</Path>
      </Paths>
      <Specs>
        <Spec>*.pdf</Spec>
        <Spec>*.jpg</Spec>
      </Specs>
    </LocalFile>
  </cmLocalViewer>
</pdroot>
```

Such a file is requesting that cmLocalViewer look in two folders (CrashData and Supplemental) for any files with the pdf or jpg extension. This is awfully general and would result in the user being presented with a long list of files to choose from. A more likely request would look like this:

```
<pdroot>
  <cmLocalViewer>
    <Command>LocalFile</Command>
    <LocalFile>
      <Paths>
        <Path>\\FileServer1\CrashData</Path>
      </Paths>
      <Specs>
        <Spec>20140702001.pdf</Spec>
      </Specs>
    </LocalFile>
  </cmLocalViewer>
</pdroot>
```

In this case, a specific file is being requested. If that file exists in the specified path, it will be opened directly without any additional user interaction.

LocalExe

LocalExe is used to execute a program on the local computer. This is useful in cases where access to scanned reports or other content is through a document management system. If that system provides a utility that can be used to connect to the repository and display a file, LocalExe is the tool to call it with.

This type of request contains the name of an .exe (program) and any parameters required by the document management utility. For example, a request might look like this:

```
<pdroot>
  <cmLocalViewer>
    <Command>LocalExe</Command>
    <LocalExe>
      <Exe>ShowReport.exe</Exe>
      <Parameters>CaseId="123" noprompt</Parameters>
    </LocalExe>
  </cmLocalViewer>
</pdroot>
```

Such a file is requesting that cmLocalViewer run a program called "ShowReport.exe" and pass it the parameters "CaseId=123 noprompt". If ShowReport.exe exists, it will be called, and will (presumably) display the crash "123" without prompt to the user. Important: note that the program does not contain a path, just the filename. This is for security purposes and is described in the security section that follows.

Security

Creating a utility that launches local files or executables simply by performing a download in a browser runs the risk of misuse by bad internet citizens. Pd' Programming has done our best to make this utility safe to install and use. The following precautions have been taken:

- When LocalFile is called with one or more paths to search, the cmLocalViewer checks its local settings to see if those paths have already been approved by the user. If they have not been approved, the utility prompts the user to authorize access to those folders when running cmLocalViewer. Once accepted, the user will not be prompted for access to that location again.
- When LocalFile is called, it also includes one or more file masks. (or an individual file name) The cmLocalViewer identifies the file extension requested and prompts the user to authorize access to that file type. Again, if authorized, that file extension is recorded and the user will not be prompted for it in the future.
- When LocalExe is called, it includes the name of a program to execute. (it does not contain a path) cmLocalViewer scans a list of authorized programs (with full paths) to see if any of them match this filename.exe. If one is found, it is considered safe and is executed with the specified parameters. If none is found, the user is prompted to locate the program. That program, with full path, is then recorded as an authorized program to be executed by cmLocalViewer.
- The lists of authorized paths, extensions and programs are stored in the Windows registry in the HKEY_CURRENT_USER section under SOFTWARE/PdMagic/cmLocalViewer. Administrators may populate these settings by policy so that the end user need not be bothered.

4.6 Data Entry

Data entry in Crash Magic is achieved using a desktop program called cmDataEntry.exe.

Adding streets

When adding a new street into the database for data entry, please be sure that the street doesn't already exist in the database.

Entering streets during data entry

Primary street: During the data entry process, if you start typing the name of the primary street and it exists in the database it will show below in the drop down box.

3 -LOCATION

On Road: Bro| Reload

City: AMBROSIA
ASHBROOK

County: BROADWAY (ACCESS) RD
BROADWAY RD

State: BROOKDALE ST
BROOKS

Country: BROOKS CIR
BROOKS ST

At Intersection: BROWN RD
BROWN RD (0)

Crossing Feature: 6

MP Num:

MP Offset:

Dir From Int: s *South*

If you try to enter a street that doesn't exist, you will get an error message upon trying to exit that field.

Cross street: During the data entry process, if you already have a primary street entered, ONLY the cross street names will show up in the drop down list.

3 -LOCATION

On Road: BROADWAY RD Reload

City:

County:

State:

Country: US

At Intersection: 1 *True*

Crossing Feature: CENTER ST 6

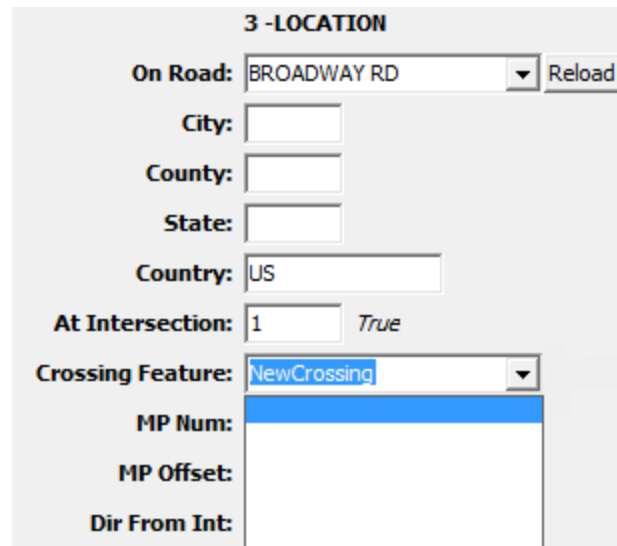
MP Num:

MP Offset:

Dir From Int:

HOWEVER, if you enter a name that already exists in the database but no previous records exist that cross the primary street, the drop down list will be blank, but the program will

allow that new street name. So you can still enter the name (because it's already in the database) that crosses that primary street.



3 -LOCATION

On Road: BROADWAY RD

City:

County:

State:

Country: US

At Intersection: 1 *True*

Crossing Feature: NewCrossing

MP Num:

MP Offset:

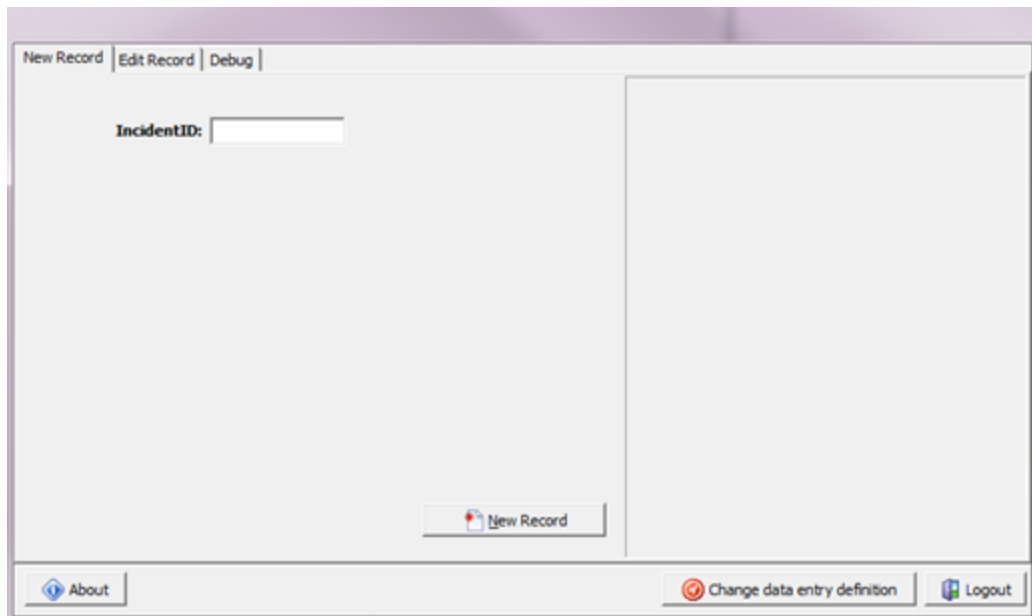
Dir From Int:

If you receive an error that the street doesn't exist, then you must add the street as a new street. (described below)

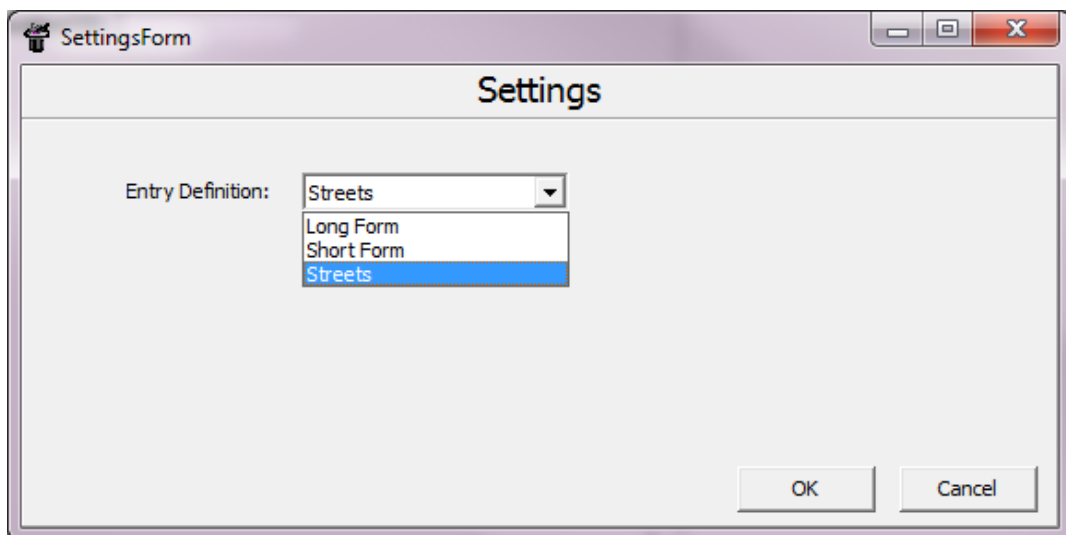
Adding a new street to the database

If you've determined that you need to add a new street to the database.

1. Log into the data entry screen, and click on the “Change data entry definition” button.



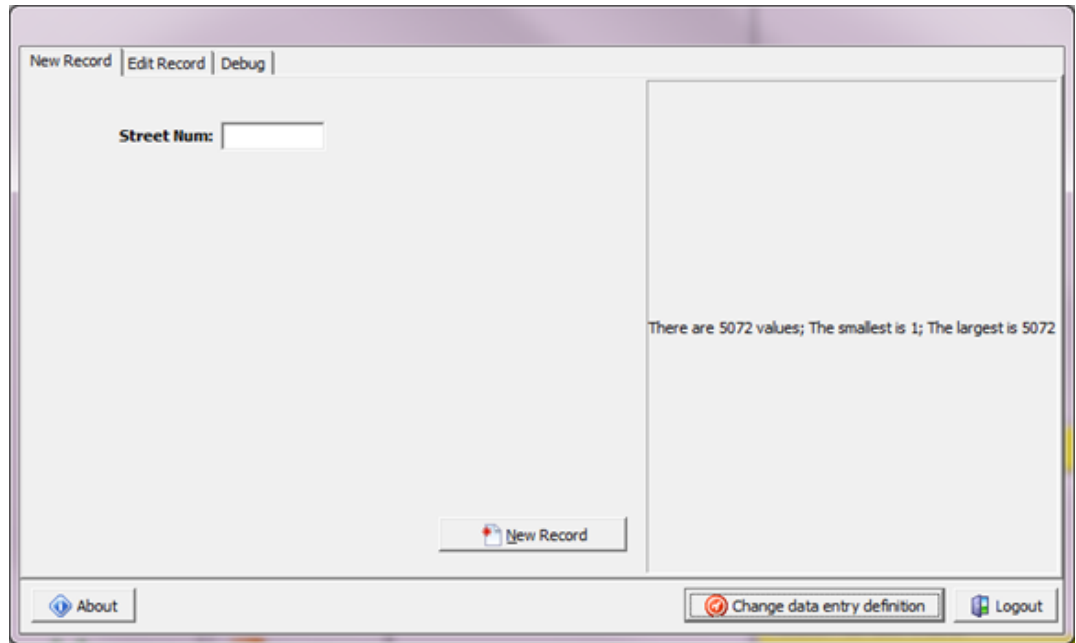
2. A screen will come up asking you to choose a new data entry definition from the drop down box. Choose the Streets option.



Then close the box by selecting the OK button.

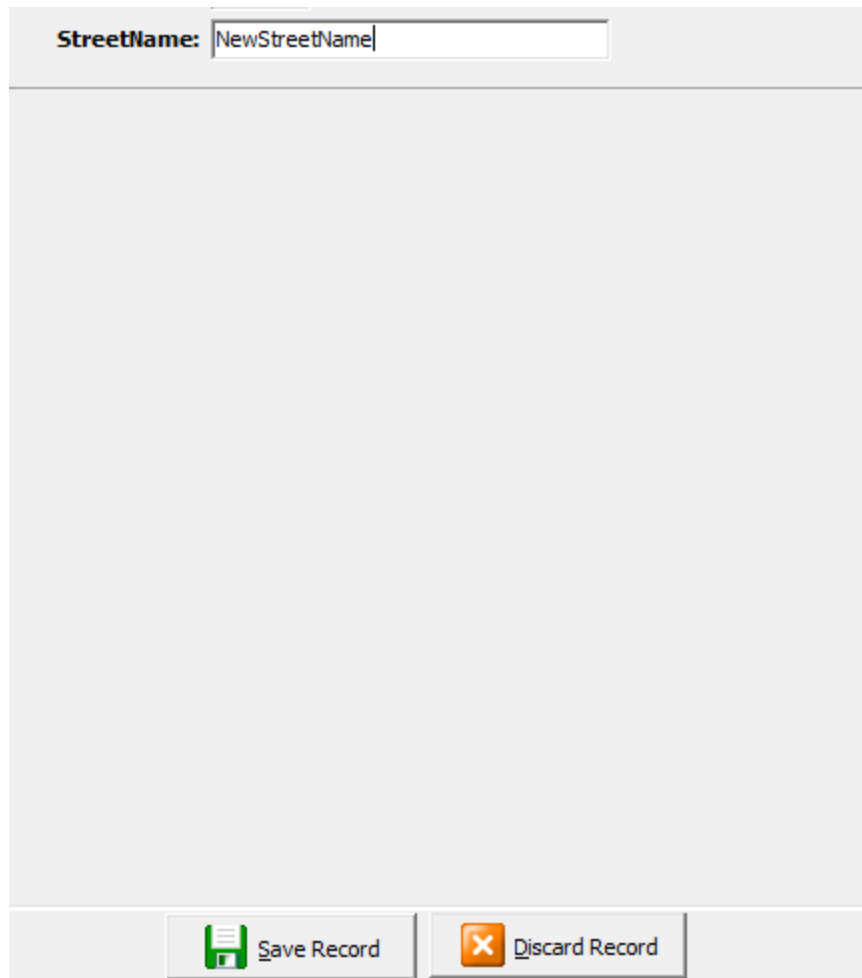
3. Next you will enter a new street number for the new street. All streets in the database are stored as numbers. You must however give it a unique number that

doesn't already exist in the database. On the right side of the screen, it tells you how many street numbers are in the database so you can choose the next available number to enter.



In this case 5072 is the largest number, so the new street number can be 5073. Enter the new number in the location and hit the "New Record" button.

4. Enter your new street name. Please make sure it is spelled correctly! Also if your jurisdiction requires a unique qualifier for the street, enter that as well.



When finished click the “Save Record” button, and you will be back to the Street Number dialog box.

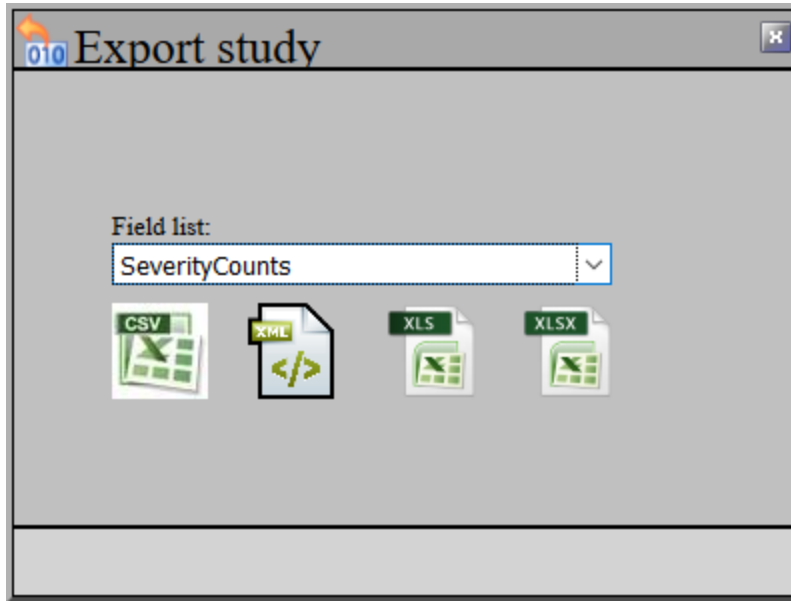
5. If you wish to enter more street names follow steps 3 and 4 again. If you are finished and want to proceed back to data entry, then click the “Change data entry definition” button again, and this time choose the short or long form for data entry.

4.7 Exporting data

Crash Magic is capable of exporting crash data in several formats. This section describes how to export data.

Export study form

The export study button  opens the export window to export collision records.



Field List: - Select from the available [field list templates](#) ¹⁰⁶ in the system. This defines the columns to be exported.

The icons below enable exporting the data as:

- CSV - ASCII comma-delimited. Can be imported into pretty much any spreadsheet/database. Unformatted except that it contains a head row with the field names
- XML - Contains the field names, data types, raw and looked up values.
- XLS and XLSX - Excel spreadsheets in native format. Fields and values are formatted as numbers/text as appropriate. *The next manual section describes how an Excel template may be used during the export.*

Note: If importing into another system, it is advised that a field list template is created that does not have spaces in the titles, this will make it more likely to be compatible with other systems.

Exporting to XLSX using a template

Normally, when data is exported using this function, the only formatting performed is data type and perhaps text alignment. However, it is also possible to use an existing

Excel spreadsheet as a template to export into. This makes possible many more types of formatting. Here are the steps for that:

1. In Crash Magic, log in as an analyst.
2. Create a field list that will be used with this new Excel template. Use a unique name when you save it. (i.e. MyFieldList1)
3. Using Excel, create a spreadsheet to be used as a template. Crash Magic data will be added to the spreadsheet starting from the A1 cell. (a default that may be changed) Perform any desired formatting.
4. Save the spreadsheet as an "Excel Template" using Excel's "Save as..." option.
5. In Crash Magic, log in as administrator. Select your own user account.
6. In the available resources box at the top of the page, choose "Excel document (exceldocument)". Press the plus key to create a new resource in your account.
7. Select "Upload a file" and choose the template you created in step 4.
8. Edit the name of this Binary content resource to be the same as that of the field list you created in step 2. (i.e. MyFieldList1)

Now, return to the analysis side of the program and export the data from your study using the field list template that you created in step 2. Choose XLSX format. When you open the provided file, it will be the result of merging your XLST template with the data from that study.

Excel document optional properties:

- Sheet name. By changing the sheet name, you can control the name of the sheet Crash Magic export will write to. If that sheet already exists, it will be used, otherwise it will be created.
- Start column, row. The default location where data will be start being written is A1. However, by setting a different start column and start row, the data will start being written there.

4.8 Filters

Crash Magic has a powerful feature called the filter. The filter enables the user to perform ad-hoc queries of the database or any current report. It can be used to gather crashes for a diagram, high crash location list, chart, etc. It can also be used to limit those crashes shown in a diagram after it has been generated.

Because the filter is tied to the study, all reports based on a particular study are therefore linked to that filter.

The filter is a true/false statement. Any time an crash is going to be evaluated, it is first checked against the current filter. If each statement in the filter is true for the given record, then the record is included, otherwise that record is excluded from the current report or diagram. Common uses for the filter is to impose a "distance cutoff" beyond which crashes are said to have "not been intersection related".

The filter can be modified by selecting filter button at the bottom of any study or report. The resulting dialog box will have space to enter a filter.

Distance < 100

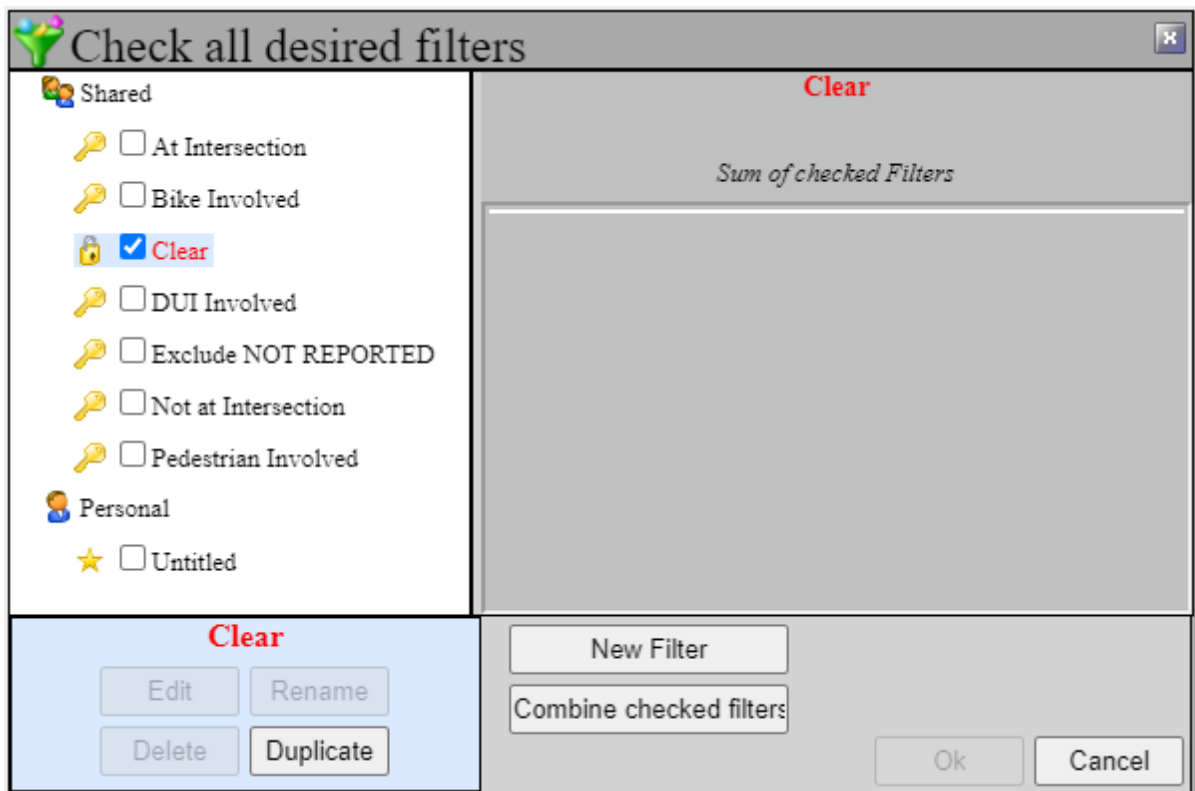
This would exclude all crashes where distance was greater than or equal to 100. The next example builds on this:

((Distance > 0) .AND. (Distance < 200)) .AND. (Type_Of_Coll = Type_Of_Coll.Rear_End)

This can be read as: Include only crashes where Distance was between 0 and 200, and Type of Collision was Rear end.

Filter Selector

The filter selector allows user to select and create filters. The tree to the left displays available filters under each owner. The section on the right displays the name and content of the filter.



New - Creates a new filter(The default name of a new filter is untitled)

Del - Deletes the currently selected filter(Only personal filters can be deleted)

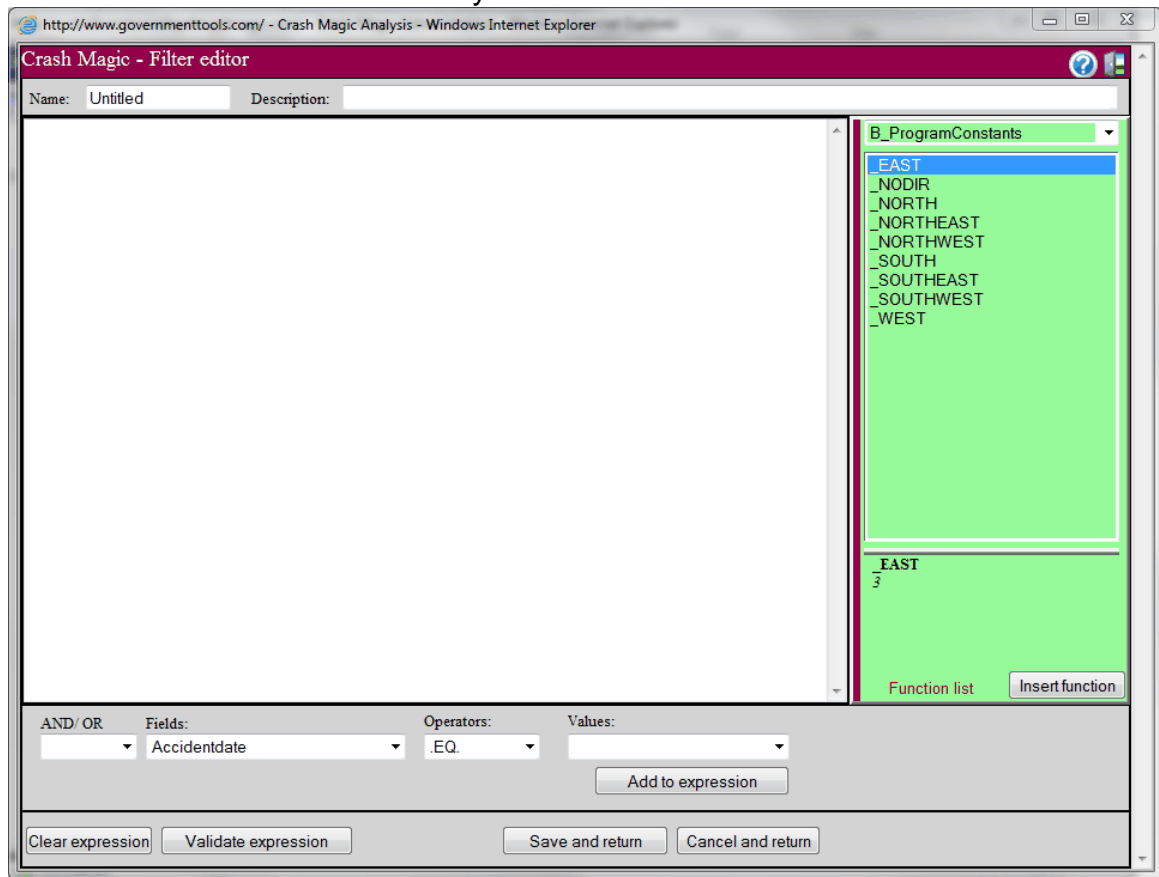
Edit Filter template - Opens the filter editor for the currently selected filter

Ok - Closes the filter selector and sets the current filter selected

Cancel - Closes the filter selector without changing the filter selected

Edit form

The filter editor window allows the current filter expression edited. Complex expressions can be quickly and easily created in the editor window. The filter editor window is divided into five sections. The Top section contains the name and description information for the filter. The left section below the name is the Filter expression editor. This expression editor allows users to type content directly into the expression. To the right of the expression editor is the function selector. The function selector allows users to select various predefined functions in Crash Magic, and view a brief summary (See [Functions](#) ⁸⁰). Below the Expression editor and function selector is the expression builder. The expression builder allows users to quickly build expressions by selecting values from the database. The very bottom section is for overall functions of the filter.



Name - Name of the current filter

Description - Description of the filter

Function category selector - This drop down menu allows users to select various function categories in Crash Magic

Function List - This link opens a popup window of standard function used in the program

Insert function - Button for inserting the current selected function into the expression editor

Expression builder:

AND/OR - This drop down adds and AND or and OR clause to the front of an expression(This is used when you have an existing expression in the editor and you want to add to it)

Fields - This drop down menu allows the user to select a field from the database to use in the expression

Operators -This drop down allow the user to select an operator for use in the expression

Values - This drop down allows the user to select a value(Values displayed are from the specific lookup information defined for the field. If a field has no lookup information this drop down menu will be blank. See [lookup queries](#)²⁷⁵ in creating a configuration)

Add to expression - This button adds the selected values from the expression builder into the expression editor

Clear expression - This button clears the expression editor section of the page

Validate expression - This button checks the expression in the expression editor window for a valid expression(Valid expressions will receive the message "Parsed successfully using the following fields/functions:")

Save and return - Saves the current filter expression returns to the previous screen

Cancel and return - Drops any changes made to the filter and returns to the previous screen

Operators

The following operators are valid for use in filter statements:

Operator:	Alternative Operator:	Meaning:
=	.EQ.	equal to
<>	.NEQ.	not equal to
>=	.GTE.	greater than or equal to
<=	.LTE.	less than or equal to
	.OR.	Or
&	.AND.	And
	.LIKE.	Like - string comparison with wild cards. Uses % as string wildcard
-		Minus
+		Plus
*		Multiply
/		Divide
#		Xor

The alternative operators result in equivilant expressions as the primary operator. They were added to the filter parser to make expressions easier to read. Primary and alternative operators may be mixed and matched within or across expressions.

Examples:

```
(( TypeOfCollision = TypeOfCollision.RearEnd ) | ( Type of Collision = TypeOfCollision.SideSwipe)) &
(Injuries >= 1)
```

is the same as

```
(( TypeOfCollision .EQ. TypeOfCollision.RearEnd ) .OR. ( Type of Collision .EQ.
TypeOfCollision.SideSwipe)) .AND. (Injuries .GTE. 1)
```

Functions

The available filter functions are grouped into several categories:

- [Date functions](#) ⁸⁰
- [Numeric functions](#) ⁸²
- [String functions](#) ⁸³
- [Schematic functions](#) ⁸⁴
- [Misc functions](#) ⁸⁴

Date functions

All date are displayed in the users localized format set in Crash Magic. The localized format will default to the localized settings of the server that is running Crash Magic if this has not been set.

Function name:

Usage:

AsDate

AsDate(aString) Returns the date type for the specified string

Examples:

AsDate("12/20/2000") = 12/20/2000 12:00 AM

BuildDate

BuildDate(DateTime, DateTime | DateTime, Time | DateTime, H,M,S |
Y,M,D,H,M,S)

Returns a localized date time for the parameters passed. Returns a null value if the time or date passed is null.

Examples:

Assume that the field aDate is a datetime field from a database = June 19, 2008

(aDate, 12/30/1899 1:33:00 PM) = June 19, 2008 1:33 PM

(aDate, 2033) = June 19, 2008 8: 33 PM

(aDate, 15, 45, 20) = June 19, 2008 3:45 PM

(2008, 9, 20, 5, 22, 2) = September 20, 2008 5:22 AM *year must use four digits*

(aDate, blank) = NULL *no value returned*

DateDiff

DateDiff(Date1, Date2 [,units = "d"])

Returns the difference, in whole numbers, between two dates in the units specified.

Valid units are y,m,d,h,n,s

DateStr

DateStr(Date)

Returns the date string of the julian number.

Day

Returns the numeric day of the month (February 5th = 5) for the specified date

DOW

DOW(Date)

Returns a number indicating the day of the week for the specified julian date. 0

represents Sunday 6 represents Saturday 7 indicates an invalid julian date

DOWStr

DOWStr(aDate | aWeekdayIndex)

Returns the string value (i.e. "Monday") day of the week for the specified date or day of week index (DOY)

DOY

DOY(aDate)

Returns the numeric day of the year (January first =1) for the specified date

EndOfMonthContaining	EndOfMonthContaining (Date) Returns the julian date of the end of the month containing the given date.
EndOfQtr	EndOfQtr (Date) Returns the julian date of the end of the quarter with the given julian date.
EndOfQtrContaining	EndOfQtrContaining (QtrNum, Year) Returns the julian date of the end of the given quarter in the given year.
FormatDate	FormatDate(DateTime, Format string) Returns the specified date formatted as indicated in a format string Assume that the field aDate is a datetime field from a database = Thursday, June 19, 2008 2:00:00 PM Format mask elements: mmm = Jun mmmm = June ddd = Thu dddd = Thursday dddd = 6/19/2008 dddddd = Thursday, June 19, 2008 hh = 14 t = 2:00 PM tt = 2:00:00 PM dd/mm/yyyy
Julian	Julian (DateString) Returns the julian of the given date string.
StartOfMonthContaining	StartOfMonthContaining(Julian Date) Returns the julian date of the start of the month containing the given date.
StartOfQtr	StartOfQtr(Julian Date) Returns the julian date of the start of the quarter with given julian date.
StartOfQtrContaining	StartOfQtrContaining(QtrNum, Year) Returns the julian date of the start of the given quarter in the given year.
StartOfStudyQtr	Takes no parameters. Returns the julian date of the start of the current study quarter.
StartOfYear	StartOfYear(YearNum) Returns the julian date of the start of the given year.
SubtractMonths	SubtractMonths(Julian, num months) Subtracts from the given julian the given num months and returns the resulting julian.
SubtractYears	SubtractYears(Julian, num years) Subtracts from the given julian the given num years and returns the resulting julian.
Today	Today takes no parameters. Returns today's julian date.
WhichQtr	WhichQtr (Julian date) Returns the quarter of the given julian.
WhichYear	WhichYear(Julian date) Returns the year of the given julian.

Database functions

Function name:

RowNum

Usage:

RowNum

Returns the row number of the current record within the current dataset.

Notes: Because SQL does not include a row number as part of its specification, this RowNum function is created by Crash Magic and has several requirements and caveats:

- The RowNum function is only provided when the query includes an "ORDER BY" clause. It does not check that the order by clause creates a unique sort order, but that is also a requirement to have a valid RowNum.

- The RowNum function, as it acts on the current dataset, does not correspond to the same record in a different dataset. It is specifically the index of that record in the current dataset only.
- If data is added to, or removed from the database that would affect the current query, the RowNum function will be refreshed and may not correspond with prior reports that used RowNum.

<FieldName>

<FieldName>

All field names for the current query are available as "calculated fields" and will be displayed alongside functions

Query_Number

Returns a value from a query. Query_Number("Query Name", "Where clause parameter:Data type="Value"; ["Where clause parameter:Data type="Value"]; [...], "", "Query field name")

Example:

Query_Number("Volume", "PrimaryStreet:String=" + LocationList.Street(1,"Name") + ";" + "CrossStreet:String=" + LocationList.Street(2, "Name") + ";" , "", "AvgVolume")

Numeric functions

Function name:

Usage:

Abs

Abs (num)

Returns the absolute value of a number.

Add

Add (num, num...)

Takes a list of numbers and adds them together. Missing data is not counted.

Avg

Avg (num, num ...)

Takes any number of values and averages them, ignoring no data.

Max

Max (num...)

Takes a list of numbers and returns the biggest one.

Min

Min (num...)

Takes a list of numbers and returns the smallest one.

Number

Number (Str)

Returns the number stored in the string, str. Returns 0 if the string is not a number.

Sqr

Sqr(num)

Returns the square of the given num.

AsString

AsString(number[, precision])

Converts a number to a string, numtrail is the number of digits following the decimal point.

This is the simpler version of AsStringF(). AsString() uses the "f" format type, no Width, and optional precision.

AsStringF

AsStringF(number, FormatString)

Converts a number to a string. FormatString looks like "%[Width][.Precision]Type

Width is the number of digits to the left of the decimal point. (spaces will be used if no significant digits are available)

Precision is the number of digits to the right of the decimal point.

Type is one of:

e = Scientific

f = Fixed

g = General

m = Money

n = Number (floating)

Examples:

```
AsStringF( 123.456, "%e" ) = 1.2345600000000E+002
AsStringF( 123.456, "%f" ) = 123.456
AsStringF( 123.456, "%.1f" ) = 123.5
AsStringF( 123.456, "%.2m" ) = $123.46
```

Note: Any text in the format string that is not part of the % specification will be added as a string literal:

```
AsStringF( 123.456, "Hello %.1f world" ) = Hello 123.5 world
```

AsStringIntF

AsStringIntF(number, FormatString)

Converts a number to a string. FormatString looks like "%[Width][.Precision]Type

This function begins by rounding the "number" value to create an integer type. The formatting is the same as AsStringF(), but the possible format types are different since the value is not floating point.

Width is the number of digits to the left of the decimal point. (spaces will be used if no significant digits are available)

Precision is the number of digits to the right of the decimal point.

Type is one of:

```
d = Decimal (integer)
u = Unsigned decimal
x = Hexadecimal
```

Examples:

```
AsStringF( 123.456, "%d" ) = 123
AsStringF( 123.56, "%d" ) = 124
AsStringF( 11, "%x" ) = B
AsStringF( 255, "%x" ) = FF
```

Note: Any text in the format string that is not part of the % specification will be added as a string literal:

```
AsStringF( 123.56, "The value is (%d)" ) = The value is (124)
```

String functions

Function name:

Concat

Copy

Length

Pos

Space

Token

Trim

Usage:

Concat (str1...)

Takes any number of string parameters, concatenates all parameters, ignoring no data.

Copy (TheStr, StartPos, Count)

Returns substring of TheStr from StartPos for Count.

Length (str)

Returns the number of characters in the string, str.

Pos(substring, string)

Returns the position of substring within string.

Space (str1...)

Takes any number of string parameters. Concatenates all parameters, inserting spaces between each.

Token(aString, aDelimiter, aIndex)

Returns a subString of aString using aDelimiter to index the string and aIndex to specify which indexed substring to return

Returns a String with any leading and trailing spaces removed

Trim(aString)

UCase Returns a string in all uppercase letters
UCase(aString)

Schematic functions

Function name:

Usage:

CorrAccLoc

Returns the current crash's location along the current diagram corridor. For a milepost corridor, this function returns the value stored in the mile_1 field. For a hundred-block corridor, this function returns the value stored in the block_1 or block_2 field depending on which street matches the primary street in corridor study. Returns the beginning of the corridor in the current study. For example, if the current study examines milepost 4.5 to 9.3, this function will return 4.5.

CorrBegin

CorrEnd

Returns the end of the corridor in the current study. For example, if the current study examines milepost 4.5 to 9.3, this function will return 9.3.

CorrLength

Returns the length of the corridor in the current study. For example, if the current study examines milepost 4.5 to 9.3, this function will return 4.8. (9.3 - 4.5)

DistGroup(0..20)

Returns the distance group values from the settings box. The first distance value is DistGroup(0), the second is DistGroup(1) and so on. These values are often used in schematics to create different areas for crashes which occurred at varying distances from the intersection.

NBArriving, NBDeparting,
SBArriving, SBDeparting, EBArriving,
EBDeparting, WBArriving,
WBDeparting

These functions perform some calculations to determine which side of the intersection an crash occurred on. The following steps are taken to determine this value:

1. Both of the vehicles' direction of travel values are examined to determine the overall direction of travel for the crash. This will indicate which function will return true. The following logic is used for this calculation:
 - a) If both vehicles were traveling in the same direction that direction is the overall direction of travel for the crash.
 - b) If only one of the directions was reported, it becomes the overall direction of travel for the crash.
 - c) If neither of the above conditions is met, then none of these functions will return true.
2. The value of _DirFromInt is examined. If such a value exists and specifies a direction from intersection for this crash, that value is used to determine whether the vehicle was "arriving" or "departing" the intersection. For example, if the overall direction of travel for the crash is Northbound and the direction from intersection is indicated to be North, then NBDeparting will return true.
3. If the no value exists for _DirFromInt, the crash is assumed to be "arriving".

North, South, East, West, NoDir

These are constants representing North, South, East and West. These constants apply to Veh1Dir and Veh2Dir

Veh1Dir, Veh2Dir

Returns the direction travel for vehicle 1 and 2 as described in the calculated fields file.

Misc. functions

Function name: Usage:

AllMatch

AllMatch(Value...)

Takes any number of parameters. Returns true if they all match the first parameter. Otherwise, it returns false

AllTrue

AllTrue(Expression...)

Takes any number of conditional expressions. Returns true if all the expressions are true. Otherwise it returns false.

AnyMatch

AnyMatch(Value...)

Takes any number of parameters. Returns true if any of the values match the first parameter. Otherwise, it returns false.

AnyTrue

AnyTrue(Expression...)

	<p>Takes any number of conditional expressions. Returns true if any of the expressions are true. Otherwise, it returns false.</p>
BinaryFileContent	<p>BinaryFileContent(FilePathAlias, FileName)</p> <p>Returns the string of a binary file Base64 encoded. The FilePathAlias is defined in the Default .options of the .shared user.</p> <p>Example:</p> <p>BinaryFileContent("SampleImages", "IMAGE_1581.jpg")</p>
DataCount	<p>DataCount (any ...)</p> <p>Takes any number of any kind of parameter and returns the number of parameters that have data (ie not null).</p>
GetProfileNum	<p>GetProfileNum (FileName, SectionName, Keyword)</p> <p>Returns a number from .INI styled file, it will return 'entry not found' if FileName, SectionName or Keyword not found.</p>
GetProfileStr	<p>GetProfileStr (FileName, SectionName, Keyword)</p> <p>Returns string from .INI styled file, it will return 'entry not found' if FileName, SectionName or Keyword not found.</p>
If	<p>If(Condition, Value1, Value2)</p> <p>If the Condition is true, this function returns Value1, otherwise it returns Value2. This function can be nested as well: If(Condition1, Value1, If(Condition2, Value2, Value3))</p>
NoMatch	<p>NoMatch(Value...)</p> <p>Takes any number of parameters. Returns true if no values match the first parameter. Otherwise, it returns false.</p>
NotNul	<p>NotNul (Value...)</p> <p>Takes any number of parameters > 1. Returns breakline if all parameters are Not NULL.</p>
NotZeroCount	<p>NotZeroCount (num...)</p> <p>Takes a list of numbers and returns the count of how many are not equal to zero.</p>
SetMatch	<p>SetMatch(Value, [...], Compare, [...])</p> <p>Takes an even number of values. Splits the list into two parts. Checks to see if there is one and only one match for each value in the list.</p> <p>Examples:</p> <p>(1,2,1,2)=true</p> <p>(1,2,2,1)=true</p> <p>(1,1,2,1)=false</p> <p>(1,1,1,1)=true</p> <p>(1,2,3,1,1,1)=false</p> <p>(1,2,3,3,1,2)=true</p>
Case	<p>Takes one parameter and a list of values. Returns the value that matches the parameter. If the parameter is not matched the default value is returned.</p> <p>The format is Case(<Parameter>,<Check>,<Return>,[<Check>,<Return>,...]<Default>)</p> <p>Example:</p> <p>Case(parameter, 1, "One", 2, "Two", 3, "Three", "Default")</p>
impRowNodeText	<p>If parameter = 1 then "One" is returned</p> <p>If parameter = 2 then "Two" is returned</p> <p>If parameter = 3 then "Three" is returned</p> <p>If the parameter does not match any value from the statement then "Default - No Match" is returned.</p> <p>impRowNodeText(aXPath, [,[default=""]][, changes=""]]</p>
	<p>Used to specify an XML element field to import. This function imports data from the name of the XML element specified.</p>

aXPath: XPath for desired node. In most cases, this will simply be the node name for the value
 default: If the node doesn't exist, this value will be returned. (typically an empty string so that a null value will be imported)
 changes: A comma-delimited list of from=to values. For example, "255=,x=12,0=" would result in values of 255 being changed to null, x to 12 and 0 to null.

Example:

```
impRowNodeText( "Veh1Dir" )
```

In this example data from the XML node Veh1Dir is imported. If the node doesn't exist, the value "" (a blank string) will be returned.

Example:

```
impRowNodeText( "Veh1Dir", "17" )
```

In this example data from the XML node Veh1Dir is imported. If the node doesn't exist, the value "17" will be returned.

Example:

```
impRowNodeText( "Veh1Dir", "", "255=,00=0,-=0" )
```

In this example data from the XML node Veh1Dir is imported. If the node doesn't exist, the value "" (a blank string) will be returned. Also, if the value of the node is 255, a blank string will be returned. And, if the value is "00", a single "0" will be returned. Finally, if the value is a dash "-", a "0" will be returned.

Examples

Generic vs User specific examples

The real power of the filter is that it can access any of the fields in the database. Some users don't know it, but every installation of the program is able to use different field names and lookup values. Crash Magic tries to keep this transparent to the user. This section has two types of examples.

Generic filter examples are ones that should work on all configurations of Crash Magic. These examples use only the base fields.

User specific filter examples are ones that require that certain fields exist in the database in order to work properly. Note that the program will warn the user if a field or lookup value does not exist. A filter will not be processed if it is not valid for the given database.

Generic filter examples

Filtering on time of day

The filter can be used to display only crashes occurring during certain times of day. To do this, the Time field is used. Time is stored as a number from 0-2400.

Show all crashes that occurred at 8am

1. Time=800

Show all crashes that occurred between 8 and 9 am

1. Time >= 800
2. Time <= 900

another way of obtaining this condition using only one line is:

1. (Time >= 800) & (Time <= 900)

Show all crashes which occurred between 7:30am and 8:30am or 4:30pm and 6pm (rush hour)

1. ((Time >=730) & (Time <= 830)) | ((Time >=1630) & (Time <= 1800))

note that extra parentheses are used to group the two time periods so that they can be compared using the | (or) operator.

Filtering on Milepost

Show all crashes that occurred between milepost 12 and milepost 15

1. (mile_1 >= 12)
2. (mile_1 <= 15)

Show all crashes that occurred between milepost 12 and milepost 15, but exclude those that happened AT milepost 12, 13, 14, and 15.

1. (mile_1 > 12)
2. (mile_1 < 15)
3. (mile_1 <> 13)
4. (mile_1 <> 14)

A single line version of this filter would be:

1. (mile_1 > 12) & (mile_1 < 15) & (mile_1 <> 13) & (mile_1 <> 14)

User specific examples

Using user fields in the filter

The filter can be used to show crashes based on the user fields in the database.

Show all crashes coded as rear end collisions

1. Type_of_Collision = Rear_End

Show all crashes that were either rear end OR broadside

1. (Type_Of_Collision = RearEnd) | (Type_Of_Collision = Broadside)

Show all rear end crashes where an injury was involved

1. Type_Of_Collision = Rear_End
2. Injuries > 0

or, if the configuration has a severity field...

1. Type_Of_Collision = Rear_End
2. Severity = Injury

Show all crashes involving a vehicle turning left and a vehicle moving straight

1. ((Mov_1=Straight) & (Mov_2=Left)) | ((Mov_2=Straight) & (Mov_1= Left))

Or by using the @ SetMatch function:

1. @SetMatch(Mov_1,Mov_2,Straight,Left)

Combining user fields and base fields

This actually takes no special arrangements, just use both types of fields.

Show all sideswipe crashes with injuries between 8am and 9am

1. Type_Of_Collision = Side_Swipe
2. Injuries > 0
3. Time >=800
4. Time <= 900

OR

1. (Type_Of_Collision=Side_Swipe) & (Injuries>0) & (Time>=800) & (Time<=900)

4.9 Logging in

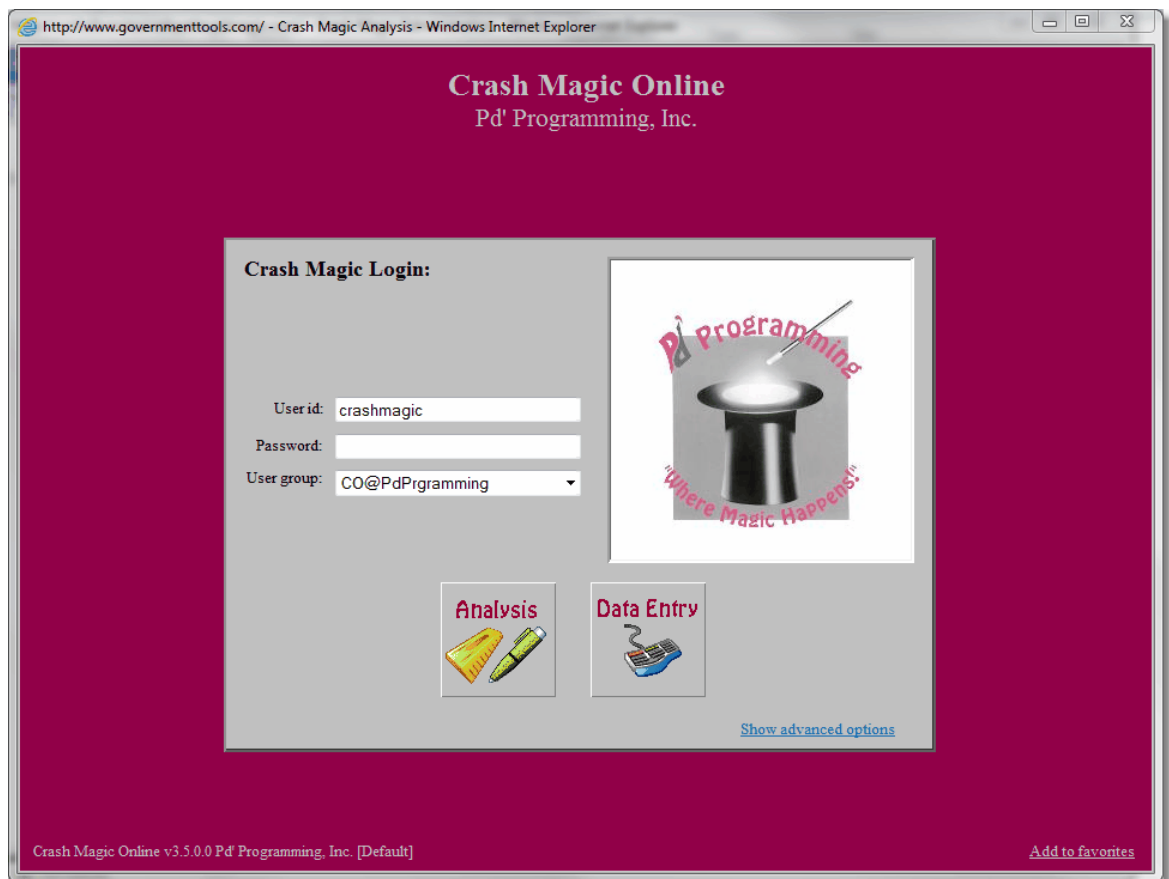
Login form

The login form is the first form you see when you access Crash Magic. It is also the last form you see upon logging out.

To login, a user id and password are required. The user id is specific to a user group, so be sure to specify the correct user group when logging in.

Once your login information is entered, you may select from one of the following buttons:

- **Analysis** - This is where most users will go. The analysis button provides access to the data gathering and reporting functions in the program. All analysis tasks are done here. This is also the default button if you merely press enter on your keyboard.
- **Data entry** - This button prompts to download the Crash Magic data entry program. This is an optional module for Crash Magic that provides access to add, modify and delete records from supported SQL databases.
- **Show advanced options** - Opens the advanced login options section of the login form



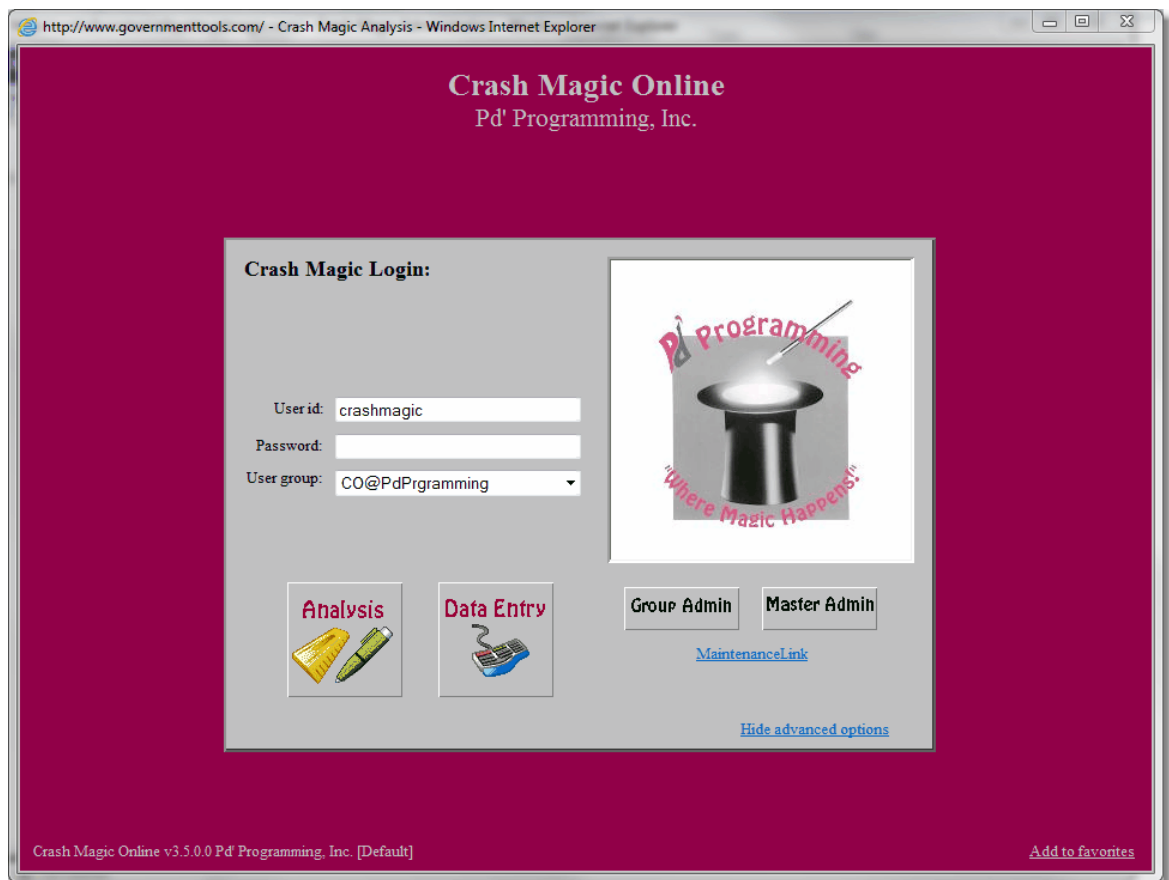
Other data shown on the login screen:

- **Close prior session on login.** This option is presented with a checkbox in the event that you failed to log out the last time you left the system. (or if you are still logged in on another computer) Only one login is possible per account, so this checkbox is provided as a convenience to terminate the other session. If you've logged in elsewhere, it can save you a trip back to that computer.
- **Database verification available for this group.** This informational text lets you know that your login can, and probably will be validated against your database login, (i.e. Oracle/MS SQL) instead of through a password stored in Crash Magic. See also (admin): [Database passthrough](#)¹⁹⁵.

- **Login into aGroup as yourName.** Crash Magic supports Active Directory logins which means that once you've logged into your computer domain, you need not log into Crash Magic as well. If your system is configured in this manner, you will rarely see the login panel. However, upon logging out, the login panel will be displayed with this prompt to let you know that you can login using Active Directory without a login or password. See also (admin): [Active Directory](#)¹⁹⁶.
- **Add to favorites.** This link, in the lower right corner, will create a proper url reference to Crash Magic in your browser. After clicking it, you will be prompted to confirm the new favorite shortcut.
- **Clear MAGICAUTO variables.** Clients that access Crash Magic from within their own in-house system, or through ArcGIS may see this message in the "Advanced" box. This link does not require any action in most cases. When another program is used to call Crash Magic, variables are passed to Crash Magic. These variables are instructions that tell Crash Magic what to do once login is complete. In some cases, it may not be desirable to perform those instructions on login. In that case, clicking on this link will delete those instructions and the user will be placed on their [home page](#)¹⁰¹ upon successful login.

The show advanced section of the login form contains the following:

- **Group Admin** - This button is used by group administrators to manage the Crash Magic system. From here, the group administrator can modify the configuration, set default reports, add or remove shared templates, and all other configuration maintenance tasks. See also: [Administration](#)¹⁴⁷
- **Master Admin** - The master admin button is used to access the administration portion of the program. Use of the master admin login should not normally be required by most users of the system.
- **MaintenanceLink** - Opens the Maintenance login window. See also: [Maintenance Form](#)²⁰⁰



4.10 Projects

In Crash Magic, a project is a container for all the settings, defaults, studies and reports that comprise the environment you work in.

Beginning users and many intermediate users will only ever need the default project entitled "Main". New projects may be created when a more organized grouping of reports is desired, or when defaults are different for a group of reports. For example, when working only with fatalities or only with pedestrian crashes. This makes it possible to have different default filters, date ranges, and even available studies or reports.

To change projects click the project button  near the upper left corner of the page. Then select a project or create a new project .

Project home page

The home page panel is the normal starting point after logging in to Crash Magic. This panel offers links to help, links to your most recent studies(s), as well as options for creating new studies.

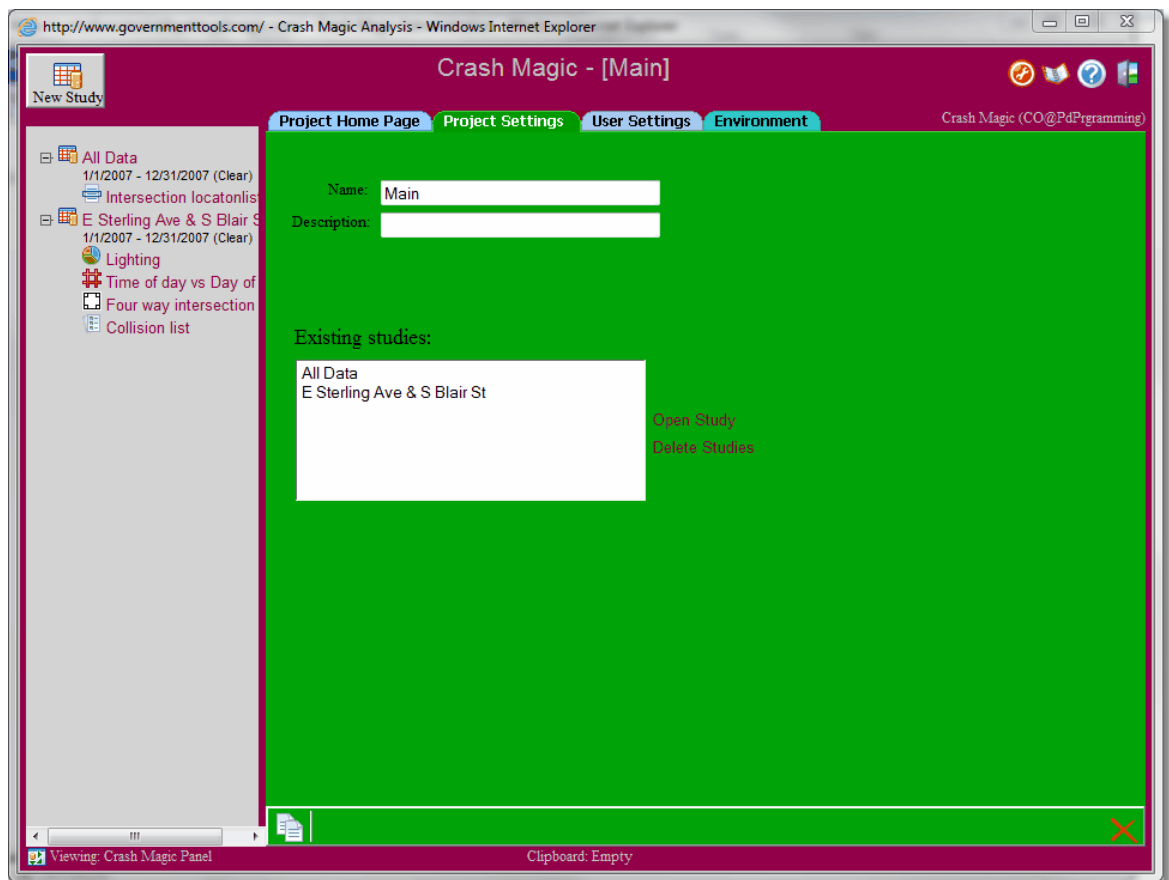


This page also contains tabs for more advanced User settings of Crash Magic

- Project Settings - This tab allows access to some of the project settings
- User Settings - This tab allows users to specify various settings and to create "Template Groups"
- Environment - This tab provides diagnostic tools for Crash Magic users
- Downloads - This tab contains download from Crash Magic. This includes the SVG plug-in to display diagrams used by version 2 of Crash Magic Online.

Project settings tab

The project settings tab allows control of various items for the project



Name - The name displayed for the project

Description - Description of the project

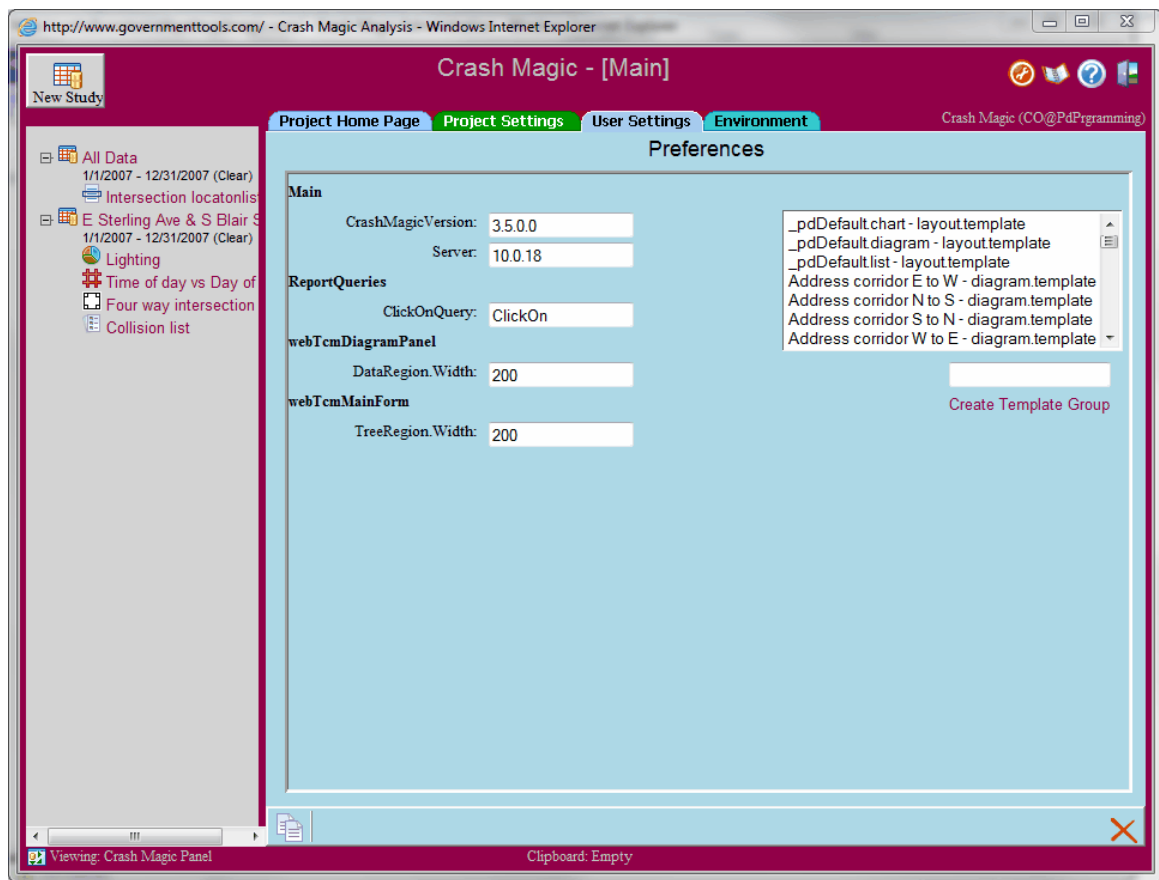
Existing studies - Displays a selectable list of studies with in the project

Open study - Opens the study selected in the existing studies list

Delete Studies - Deletes the studies selected in the list of existing studies

User settings tab

The user settings tab controls user display items



Main

Crash Magic Version: displays the current version of Crash Magic

Server: - displays the database server version that Crash Magic is connected to

Report Queries

ClickOnQuery: Displays the name of the ClickOn query

Diagram Panel

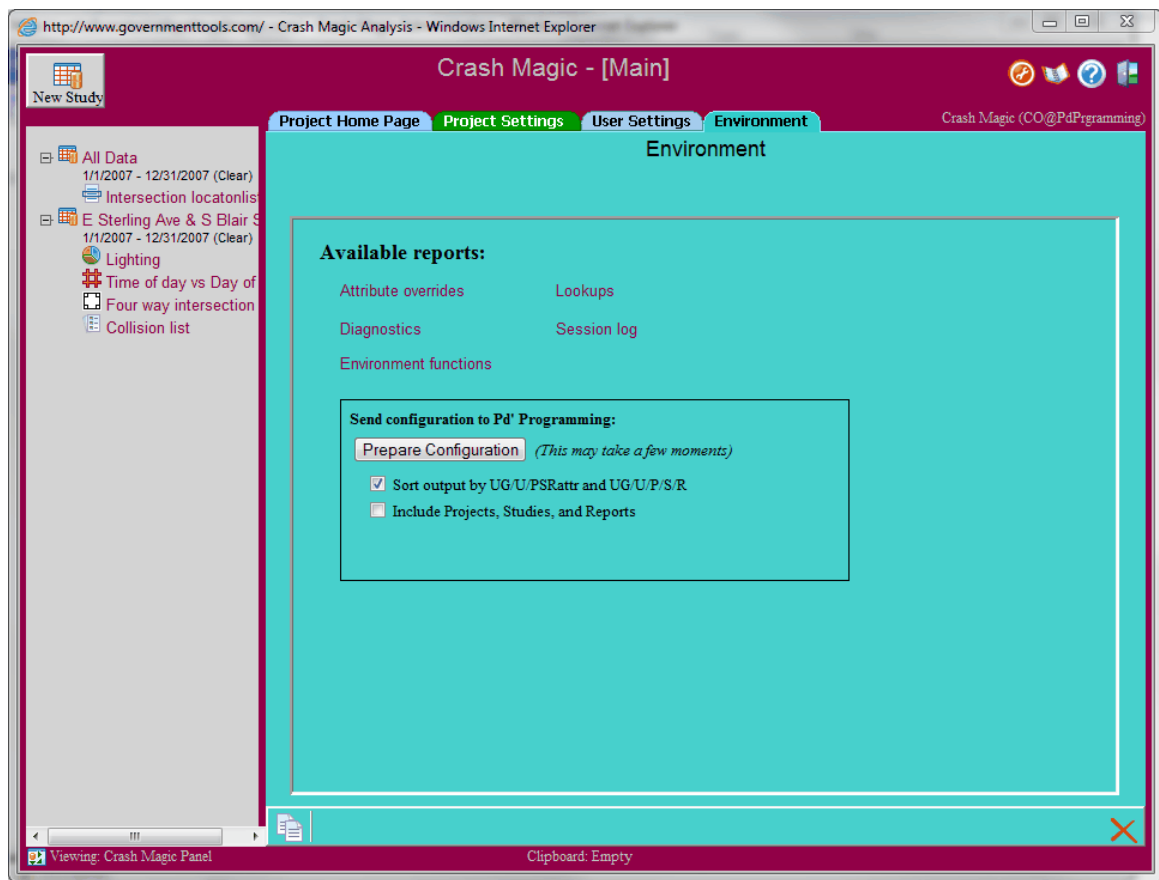
Data Region Width: Displays the current width of a diagram report

Main Form

Tree Region Width: Displays the current width of the project tree

Environment tab

The environment tab provides additional information on Crash Magic



Attributes overrides - Displays a popup report of configuration items with highlighted items indicating the objects have been overridden by the user group or user.

Diagnostics - Displays a popup report of diagnostic configuration information

Environment functions - Displays a popup report of common functions used by Crash Magic

Lookups - Displays a popup report of the current lookup information for the configuration

Session log - Displays a popup report of the current user session log

Prepare Configuration

Sort output by UG/U/PSRattr and UG/U/P/S/R

Include Projects, Studies and Reports

Submit to Pd'

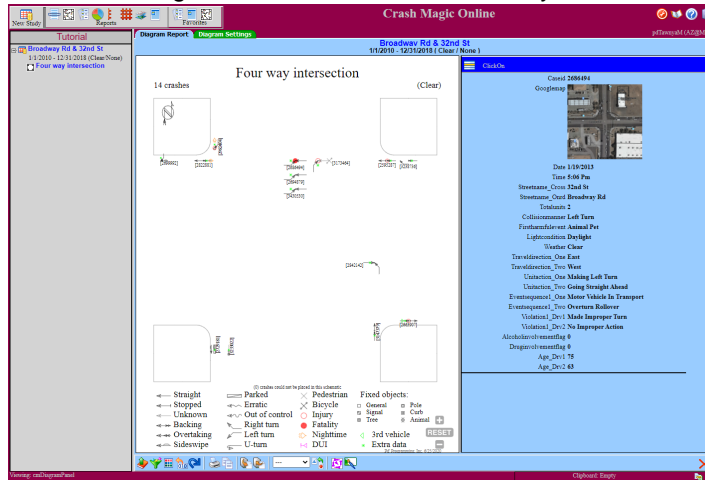
4.11 Reports

Under each study, there can be any number of reports. A report is information displayed based on the study. A simple report might show the number of crash records found in a query. A more complex report might calculate the cost of the crashes in a study.

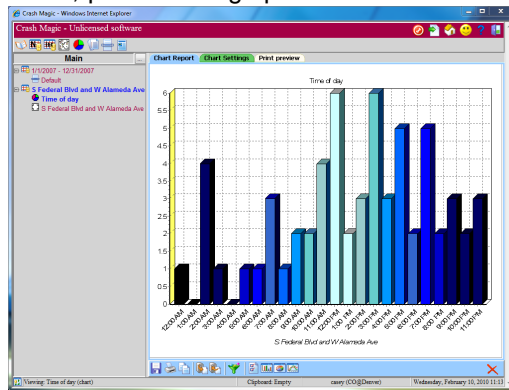
Crash Magic is capable of producing many types of reports. It is also possible to create customized reports as "plug-ins" to Crash Magic.

The "built-in" reports that come with Crash Magic include:

- Collision diagrams with "drill-down" to identify individual crash records



- High crash location lists
- Bar, pie and line graphs



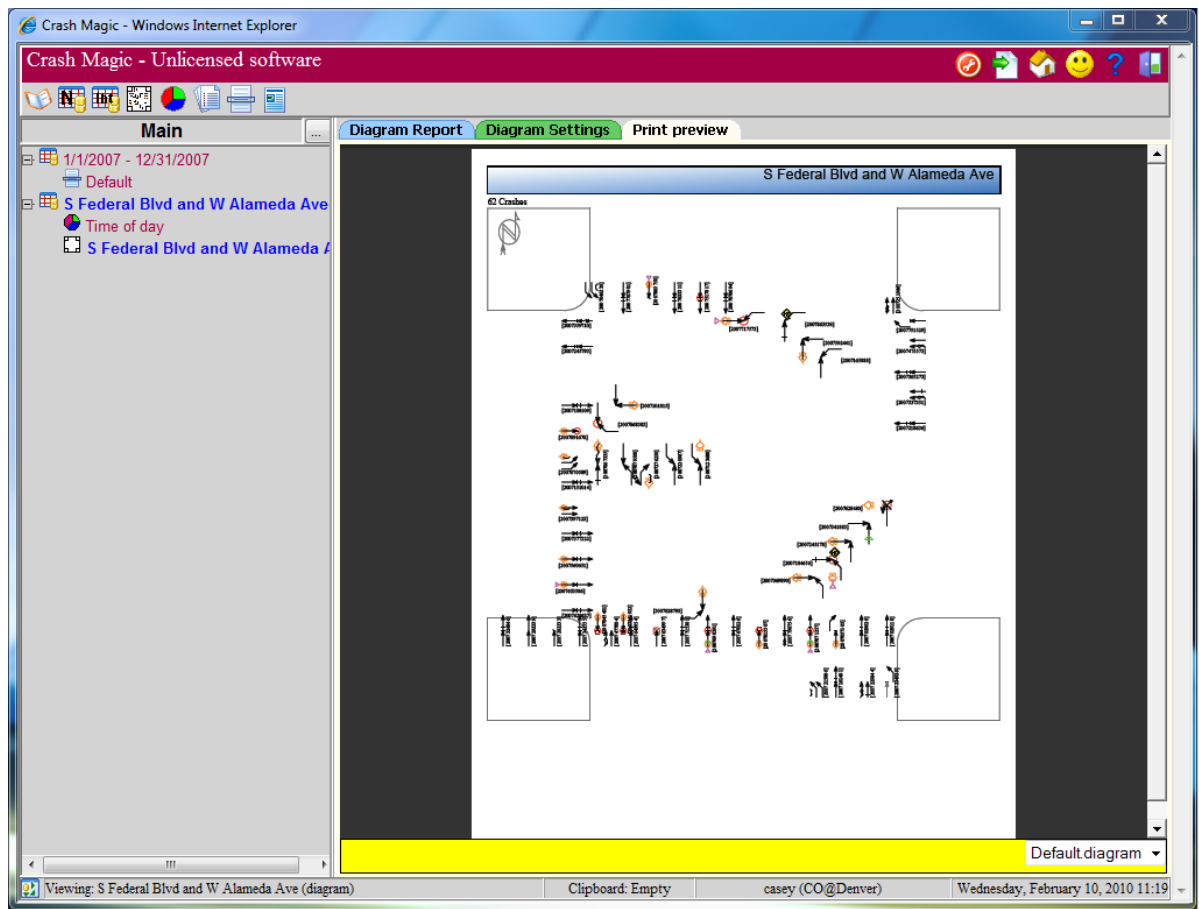
- Crash listings

Crash Data						
Date	Time	Lat	Lon	Vehicle	Injury	Message
1-15-2010	10:18 am	39.74	-104.98	0	No	ONE MOTOR VEHICLE - GOING STRAIGHT
1-23-2010	1:47 pm	39.74	-104.98	0	No	SAME DIRECTION - BOTH GOING STRAIGHT -
1-23-2010	1:47 pm	39.74	-104.98	2	No	SAME DIRECTION - BOTH GOING STRAIGHT-REAR
1-23-2010	6:41 am	39.74	-104.98	0	No	ONE MOTOR VEHICLE - GOING STRAIGHT
2-11-2010	8:30 am	39.74	-104.98	0	No	ONE MOTOR VEHICLE - GOING STRAIGHT
2-11-2010	11:51 am	39.74	-104.98	1	No	ONE MOTOR VEHICLE - GOING STRAIGHT
2-21-2010	3:23 pm	39.74	-104.98	0	No	SAME DIRECTION - BOTH GOING STRAIGHT -
2-22-2010	6:20 pm	39.74	-104.98	0	No	SAME DIRECTION - ONE STRAIGHT ONE STOPPED
2-6-2010	11:51 am	39.74	-104.98	0	No	OPPOSITE DIRECTION - ONE STRAIGHT ONE LEFT
2-8-2010	6:10 am	39.74	-104.98	0	No	ONE MOTOR VEHICLE - GOING STRAIGHT
3-12-2010	3:50 pm	39.74	-104.98	0	No	SAME DIRECTION - BOTH GOING STRAIGHT -
3-20-2010	7:50 am	39.74	-104.98	0	No	SAME DIRECTION - BOTH GOING STRAIGHT -
3-20-2010	9:10 am	39.74	-104.98	0	No	ONE MOTOR VEHICLE - GOING STRAIGHT
3-20-2010	7:29 am	39.74	-104.98	0	No	ONE MOTOR VEHICLE - GOING STRAIGHT
3-20-2010	6:13 am	39.74	-104.98	0	No	SAME DIRECTION - BOTH GOING STRAIGHT -
3-20-2010	6:38 am	39.74	-104.98	1	No	ONE MOTOR VEHICLE - GOING STRAIGHT
3-7-2010	6:17 pm	39.74	-104.98	0	No	SAME DIRECTION - ONE STRAIGHT ONE STOPPED

Whenever a report is created, a new node is added to the project tree to hold the settings for that report.

The report content itself is not stored. This is important to note. If a report is created, and the data behind the report changes, the next time that the report is displayed, it will reflect the changed data.

To save a report as it exists today, so that its contents will not change in the future, select the printer icon at the bottom of any report page and save the resulting Adobe Acrobat PDF file to your hard disk.

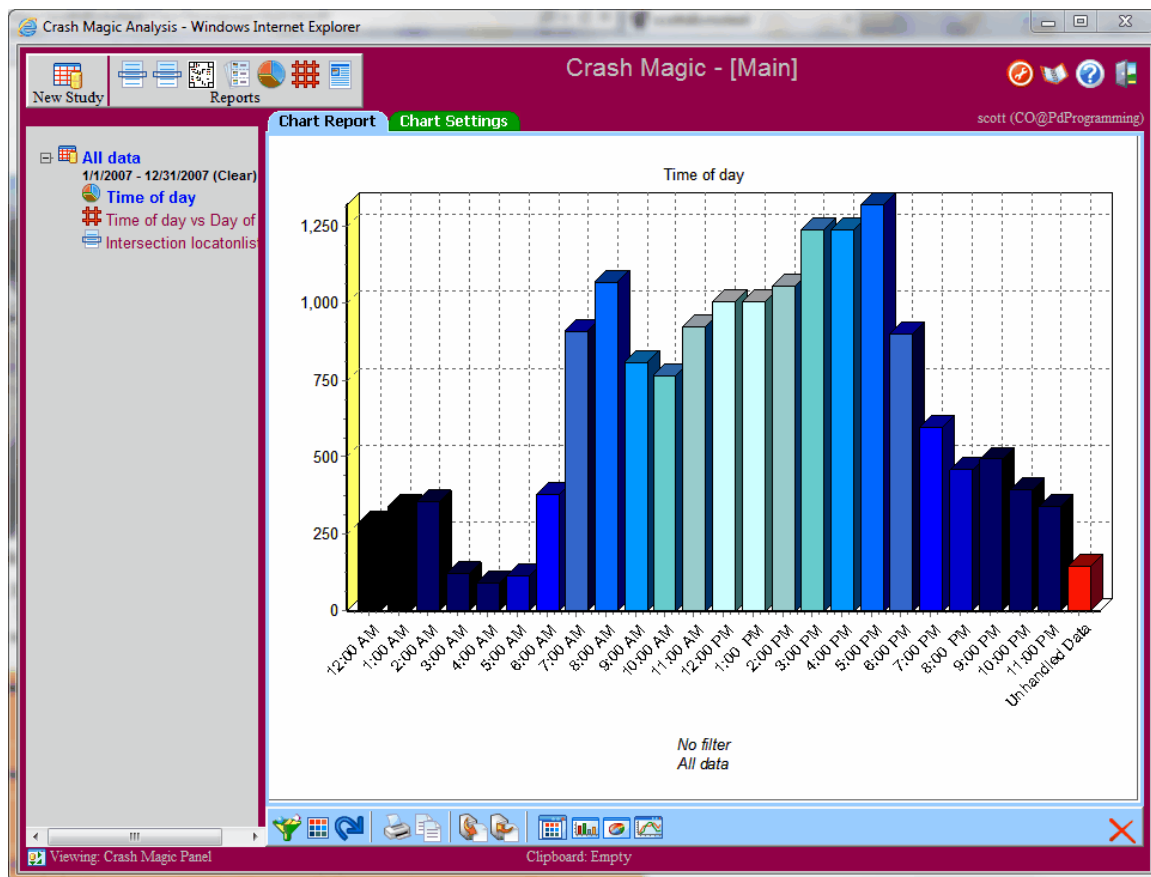


Charts





The chart panel allows you to create customized charts based on the current study. This section will explain how the chart panel and editor works.

Chart report tab

The chart report panel displays chart information based on data from the study the chart report is under.



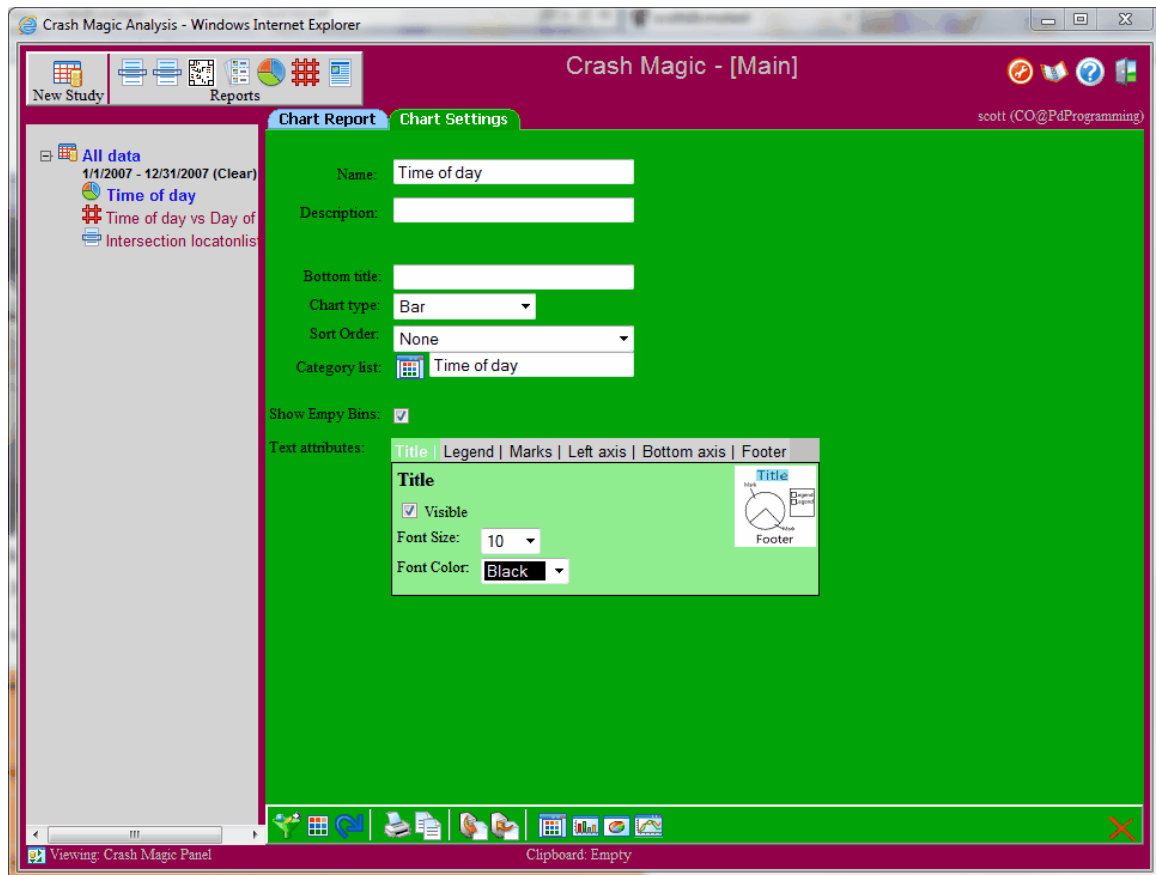
Along with the [common function buttons](#)^[66] at the bottom of the panel the following buttons can be found.

-  - Opens the [category list selection window](#)^[62] for the chart
-  - Change to bar chart
-  - Change to a pie chart
-  - Change to a line chart

Depending on the field that was selected, there may be an extra bar in your chart labeled "Unhandled Data". This is data that does not have a bin associated with the value. As an example a chart created from a database with only four lighting conditions(1 "Daylight", 2 "Dawn or Dusk", 3 "Dark - Lighted", and 4 "Dark - Not Lighted"), and contains a collision record with a lighting condition of 5. The program is not able to determine a bin for a value of 5, and the collision record would be counted as "Unhandled Data".

Chart settings tab

The chart settings tab controls the display elements of the chart.



- **Name** - Name of the chart that will be displayed in the project tree, and as the title of the chart.
- **Description** - Description of the chart
- **Bottom Title** - The bottom title is a piece of text that appears at the bottom of the chart.
- **Chart Type** - Choose from Pie, Bar, Line, Area, and horizontal variants.
- **Sort Order** - None will show the bins in the order that they are listed in the chart editor. Name will sort the bins A to Z. Count will sort the chart from least to most crashes.
- **Category list** - Opens the [category list selection window](#)⁶² for the chart. The category list is the group of bins that will be used by the chart.
- **Show Empty Bins** - When checked chart bins with 0 crashes will still show up in the key and as a blank item on bar charts. When unchecked these bins will be hidden.
- **Text Attributes** - Every text property on the chart can be made visible or hidden. They can also have their font size and color adjusted.

Layout Elements

Chart.FO (PSRO,FormatSpec)	Returns a chart to be placed inside of a FO document.	
	FormatSpec:	
	Width	The width of the chart with units. <i>Default: 10cm</i>
	Height	The height of the chart with units <i>Default: 5cm</i>


Collision diagrams

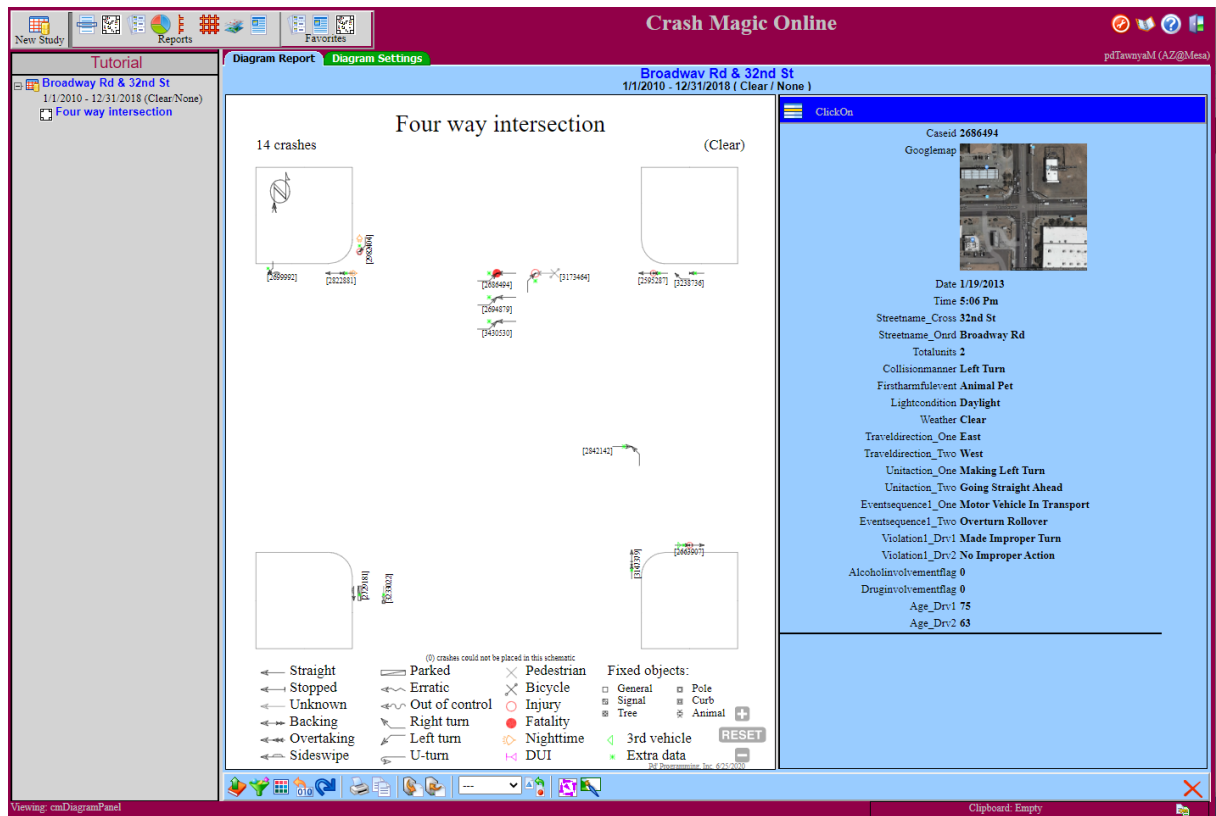
A collision diagram displays standard and custom symbols that describe each of the crashes that occurred at a specific location over a specified period of time. In order to generate a collision diagram, a study must be selected. (a collision diagram can also be generated if another report in a study is selected)

The collision diagram is created based on configurable rules defined in the administration portion of the program. These rules define how to represent the "schematic" of the location, how to build each of the crash graphics, and where to place each crash graphic within the selected schematic.

Diagram report tab

The diagram report panel displays a collision diagram and (optionally) data from a single clicked-on crash graphic. The [settings panel](#)¹⁰⁰ provides access to the diagram's display options.

Clicking on the field list icon  at the top of the panel opens the [field list editor](#)¹⁰⁶. The field list editor allows users to change the fields displayed. Clients that store images in their database can open the drop down menu to select the ClickOnImages fields.



Along with the [common function buttons](#) ^[66] at the bottom of the panel the following buttons can be found.



- Add annotations to the diagram.



- Delete the current diagram, and move to the study.

Diagram settings tab

The diagram settings panel enables you to change a number of diagram display settings.

- **Name** - displayed at the top of the diagram.
- **Description** - used for tooltips in the project tree.
- **Spacing** - defines the distances between crashes in the same group.
- **Pen width** - defines how thick the crash graphic lines should be drawn.
- **Show title** - determines if any header information will be displayed above the diagram.
- **Display legend** - determines if the key will be shown at the bottom of the diagram.
- **Schematic** - determines which [schematic](#)^[102] will be used to locate the crash graphics and curb lines.
- **Background** - specify a physical file or url to display a raster image "behind" the diagram. (i.e. ortho photo).
- **Label color** - pick a color for the labels.
- **Label content** - an expression that resolves to text which will be displayed next to each crash graphic. (can also simply be a field name).
- **Label size** - height of the label text. (the diagram is 1000 units wide)
- **Object Map** - The current object map used to display the collision graphics on a diagram.
- **Primitives** - The current list of graphic objects used to create the object map.
- **Show areas** - for debugging and modifying schematics. Displays the areas where graphics will be placed, and a graduated grid for aligning them.
- **Show Grid** - Displays a grid on the diagram schematic.

- **Reset Crashes** - Returns collision graphics to their original placement.
- **Remove Annotations** - Removes all annotations added to the diagram.

Schematics

A schematic is what describes the appearance of a diagram. To represent a 'T' intersection, you select one of the 'T' schematics. To represent an offset intersection, you choose an offset schematic. If your data contains information about lanes, turn bays, distance from intersection, boulevards, mid-block crashes, etc. there is probably a schematic to represent your data.

Schematics are selected on the settings tab of the diagram panel. A schematic contains the graphic representation of the curb lines and other physical attributes of the intersection. More importantly, it also contains the description of a number of 'areas' where crashes can be located. Each of these areas includes a condition, much like a filter, which describes which crashes can be located there. Thus, a schematic can be created to suit almost any location you can describe. This also allows you to make use of fields in your database that is unique to your data.

This chapter describes the available schematics, when they work, what they show, how to select them, and then how to modify or even create new schematics.

The default schematic that will be used every time a diagram is generated is a four-leg intersection that is capable of representing crashes at varying distances and directions from the intersection. This default can be changed at any time by choosing Utilities / Configuration Defaults / File names and clicking the 'schematic' button.

A different schematic may be selected for a diagram by choosing it from the diagram panel settings tab.

The 100 or so available pre-defined schematics are arranged by type of location. A number of different sub categories provide representation of different directions, as well as lanes.

The basic pre-defined categories are:

- **Address** – These schematics are dynamically generated based on the address range of the study. Each section of the schematic represents a hundred block range. Collisions are then placed based on the value from the `_BlockAddress` field in the `P_Default` calculated fields. The `_BlockAddress` field will be set to 0 for collision data that does not contain address information. The two characters at the end of Address indicate the direction of the block range.
- **Approach** – These schematics represent just one leg of an intersection. (i.e. the north approach) These schematics provide a high degree of detail related to the distance from the intersection. Since more room is available for crash graphics in the approach leg than in a 4 leg diagram, more areas are defined. Collision data must have distance from intersection and direction from intersection defined in the `_DistFromInt` and `_DirFromInt` of the `P_Default` calculated field. Collision data that lacks this information will default to 0 distance from intersection and a null direction from intersection.
- **Corridor** – These schematics are designed to represent crashes that contain address, hundred block data or other reference location information. They can depict the relative position of each crash along the given corridor. A number of different corridor options provide greater resolution / spacing between the end points.
- **Corridor/Part/** – These are various sections of the Address, Corridor and Milepost schematics. Dynamic schematics are built from these parts. Users should avoid selecting these schematics.

- **Int_4** – These are the most commonly used schematics. They depict 4 leg intersections of many types. Intersections with 1, 2, 3 and 4 turn bays. There are also boulevard schematics and schematics which provide special support for distance and direction from the intersection.
- **Int_T** – These are ‘T’ intersections. They are available in most of the same varieties as the Int_4 category. As would be expected, they are available for each approach direction.
- **Int_Y** – These are ‘Y’ intersection templates. In order for them to be used, they must first be customized to reflect the data available that indicates which street(s) the crash occurred on.
- **Mid_blk** – These schematics depict locations which include crashes which occurred between intersections. There are a variety of these which can also include lanes when such data is available.
- **Milepost** These schematics are dynamically generated based on the milepost range of the study. Each section of the schematic represents a tenth of a mile section. Collisions are then placed based on the value from the `_Milepost` field in the `P_Default` calculated field. The `_Milepost` field will be set to 0 for collision data that does not contain milepost information. The two characters at the end of Milepost indicate the direction of the range.
- **Mult_int** – These are diagram template(s) that depict more than one intersection in the same diagram. These will need to be modified to support data.
- **old** – This category contains all of the pre-version 6.0 schematics. Since the schematics changed fairly substantially in version 6, and some users may be attached to the original ones, we’ve provided updated versions of the original schematics.
- **RoundAbout** – These schematics are designed to display roundabout intersections.

Scanned crash reports

Crash Magic can display scanned images when a specific crash has been selected within a diagram. The administrator who creates the query that you use can specify certain fields in the database as image fields and give them an image type. If these fields are defined, they can be selected in the field list editor for listings and click-on reports in the diagram. Rather than showing the value of that field in the list, a link will be created that will display that scanned report.

Layout Elements

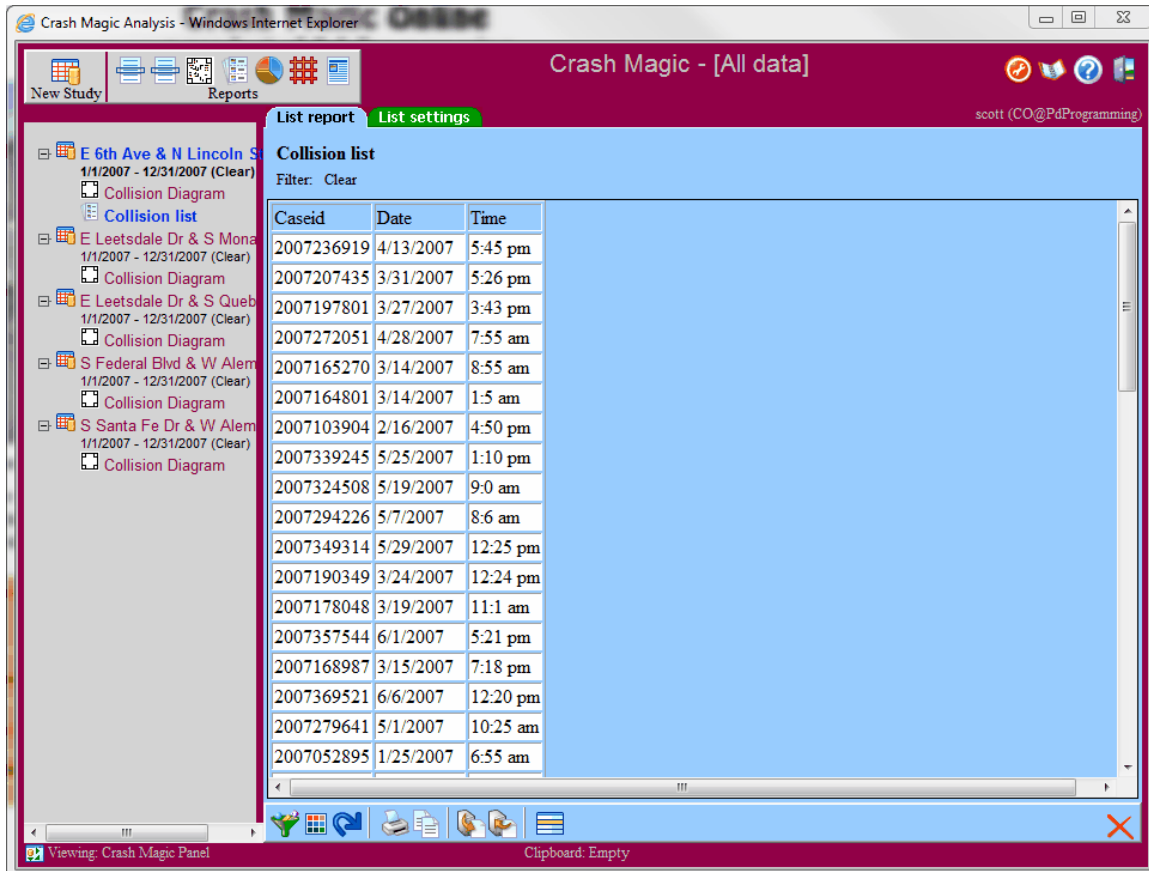
Diagram.FO(PSRO,FormatSpec)	Returns a diagram to be placed inside of a FO document.	
	FormatSpec:	
	Width	The width of the diagram with units. <i>Default: 10cm</i>
	Height	The height of the diagram with units <i>Default: 10cm</i>
	Show Key	True/False value that turns the key on and off <i>Default: true</i>
	Show Title	True/False value that turns the title, count, and filter at the top of the diagram on and off. <i>Default: false</i>

Crash listings

The list panel allows users to create a list of collision record fields for display. Users can choose a the specific fields to display and sort the information in multiple ways.


List report tab

The list report displays collision record fields selected by the user

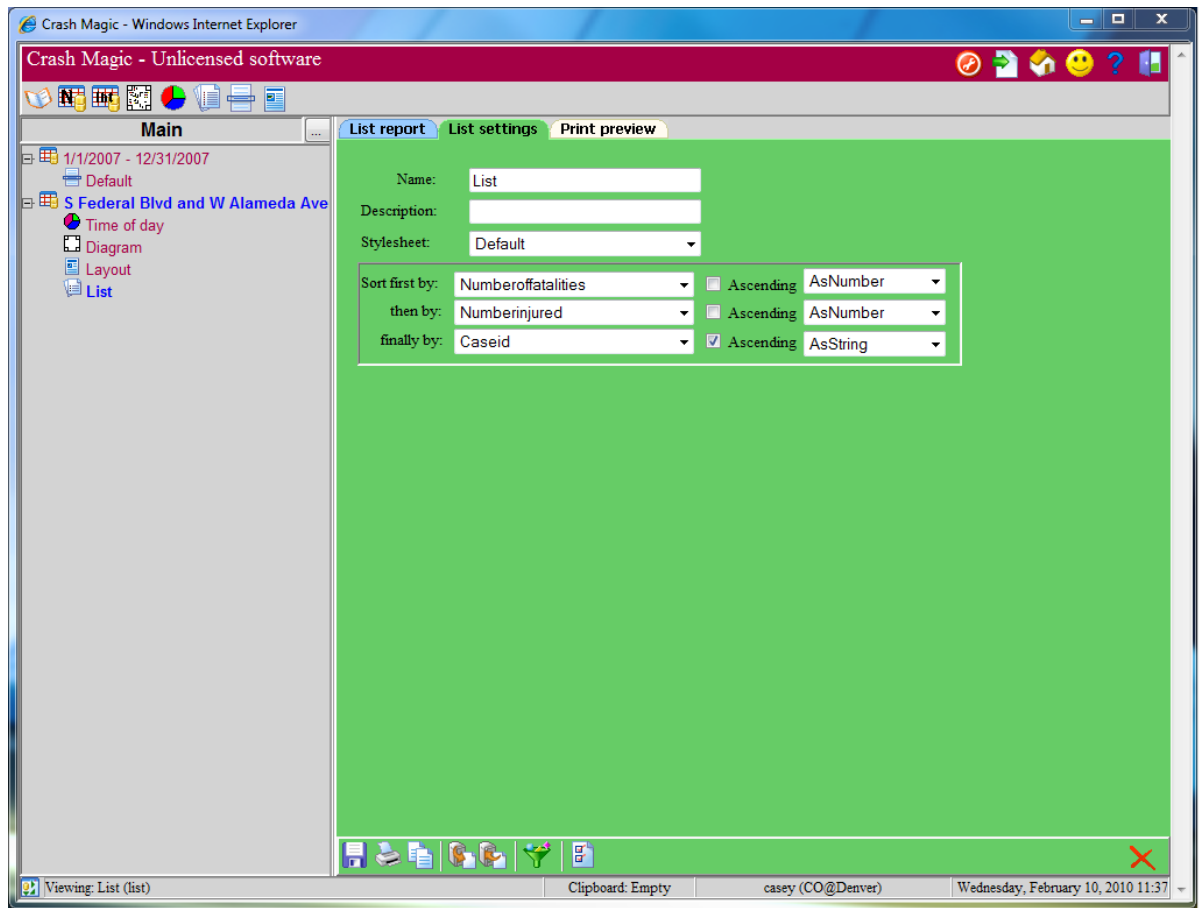


Fields show are examples of the Caseid field list in the .master .shared user. Your fields may be different.

Along with the [common function buttons](#)^[66] at the bottom of the panel the following button can be found.

-  - As with the click-on report this button will open the [field list selector](#)^[105]. This allows users to select the fields for display.

List settings tab



Name - The name that will show on the list report

Description - The description of the report

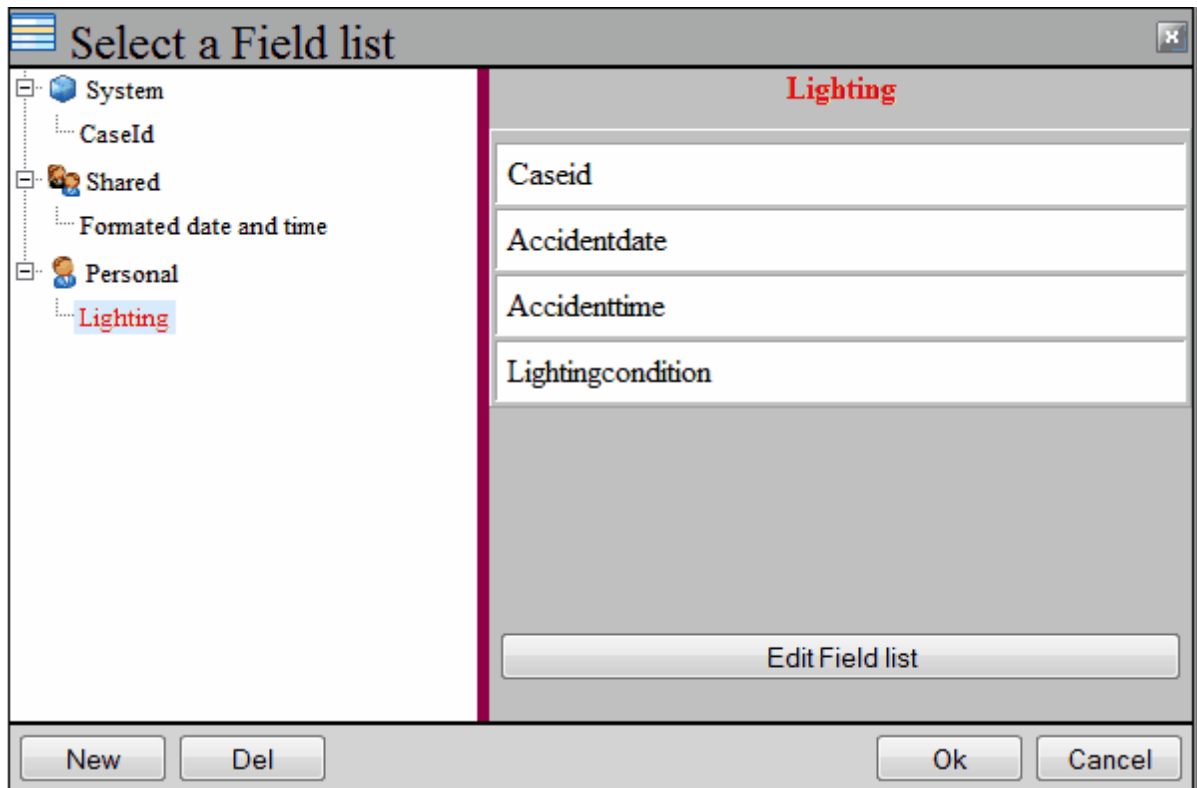
Stylesheet - The stylesheet will change how Crash Magic displays the listing.

Sort - Listings can currently be sorted by three fields. After picking the field name, you need to pick the data type so the listing knows to sort number fields correctly.

Ascending - Changes the order of the sort lowest to highest vs highest to lowest

Field list selector

The field list selector allows the user to choose a field list or create their own field list.



New - Creates a new blank field list called untitled

Del - Deletes the currently selected field list

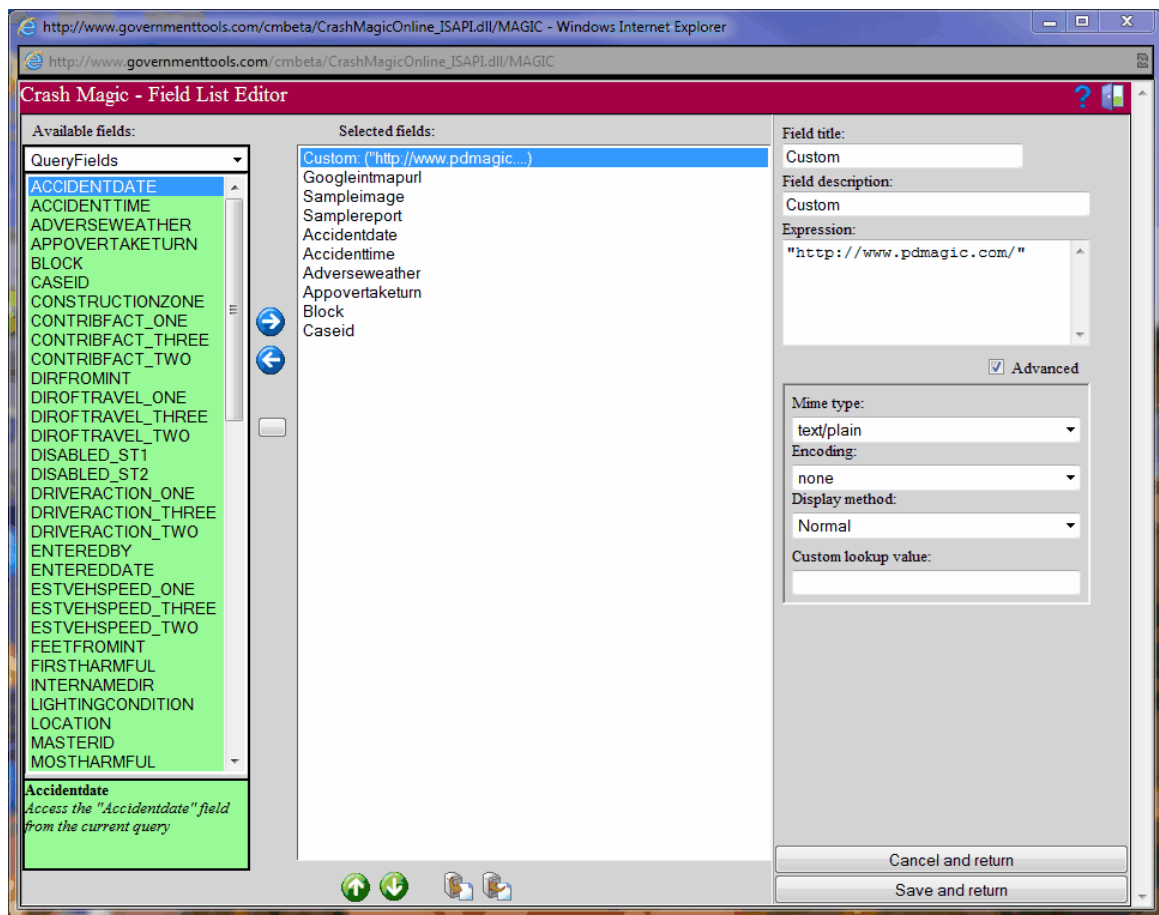
Edit Field list - Opens the [field list editor](#)¹⁰⁶ for the selected field list

Ok - Closes the field list selection list box and sets the category list to the current category list selected








Cancel - Closes the category selection list box without selecting the category list

Field list editor

The field list editor allows the client to determine the list of fields from the query to display.



The QueryFields displayed are specific to the specific collision database in use. Your fields may be different.

- **Available fields** - These are the various field list available to the Crash Magic user. Changing the drop down menu will change the list of fields displayed. The default list displayed is QueryFields. QueryFields contains the list of database fields specific to the client collision database. Fields shown in the list below the drop down can be selected for display.
- **Selected fields:** - These are the current fields selected to display by the user.
-  - Adds selected fields from the Available fields list to the Selected fields list.
-  - Removes selected fields from the Selected fields list. Removing the selected field does not delete any data. It only hides the field from the display.
-  - Allows users to create custom fields based on expressions.
-  - Move selected field or fields up in the list of Selected fields to be displayed. This allows the field list order to be changed.
-  - Move selected field or fields down in the list of Selected fields to be displayed.
-  - Load field list [template](#)¹⁴³.
-  - Save field list [template](#)¹⁴³.

One a field is selected in the Selected fields list users can edit the properties of the field.

- **Field title:** - This is the title of the field that will be shown in the report.
- **Field description:** - This is the description of the field that will will be shown.
- **Expression:** - This is the expression used to display the field. Users are not limited to display of only the field. Complex expressions can be created for display. For example the following code will display 1 Million for fatal collisions, 25 Thousand for injury collisions and 5 Thousand for property damage only.

```
IF(_IsFatality,
    "1 Million",
    IF(_IsInjury,
        "25 Thousand",
        "5 Thousand"))
```

- **Advanced** - Opens the field display settings window.
- **Mime type:** - This drop down menu allows users to tell Crash Magic how the field should be handled based on the MIME standard.
- **Encoding:** - This drop down is used when displaying image fields to tell Crash Magic the encoding for the image.
- **Display method:** - This drop down is also used when displaying an image. Users can choose to display a thumbnail of the image, or a link to the image.
- **Custom lookup value:** - This allows the user to specify a different lookup field name. The QueryFields list of fields by default are looked up values. Crash Magic converts database values to readable values for display. As an example a field of LightCondition may contain values 1 through 5 for the various light conditions. Crash Magic queries the database lookups for this field to display ("Daylight", "Dark", etc) for each of the numeric values. Users can specify a database lookup field that may contain different lookup values like ("Full Sunlight", "Dark no moon light", etc) for the values.
- **Cancel and return** - Returns to the previous page with no changes to the list of fields.
- **Save and return** - Saves the changes made to the list of displayed fields and returns the user to the previous page.

Layout Elements

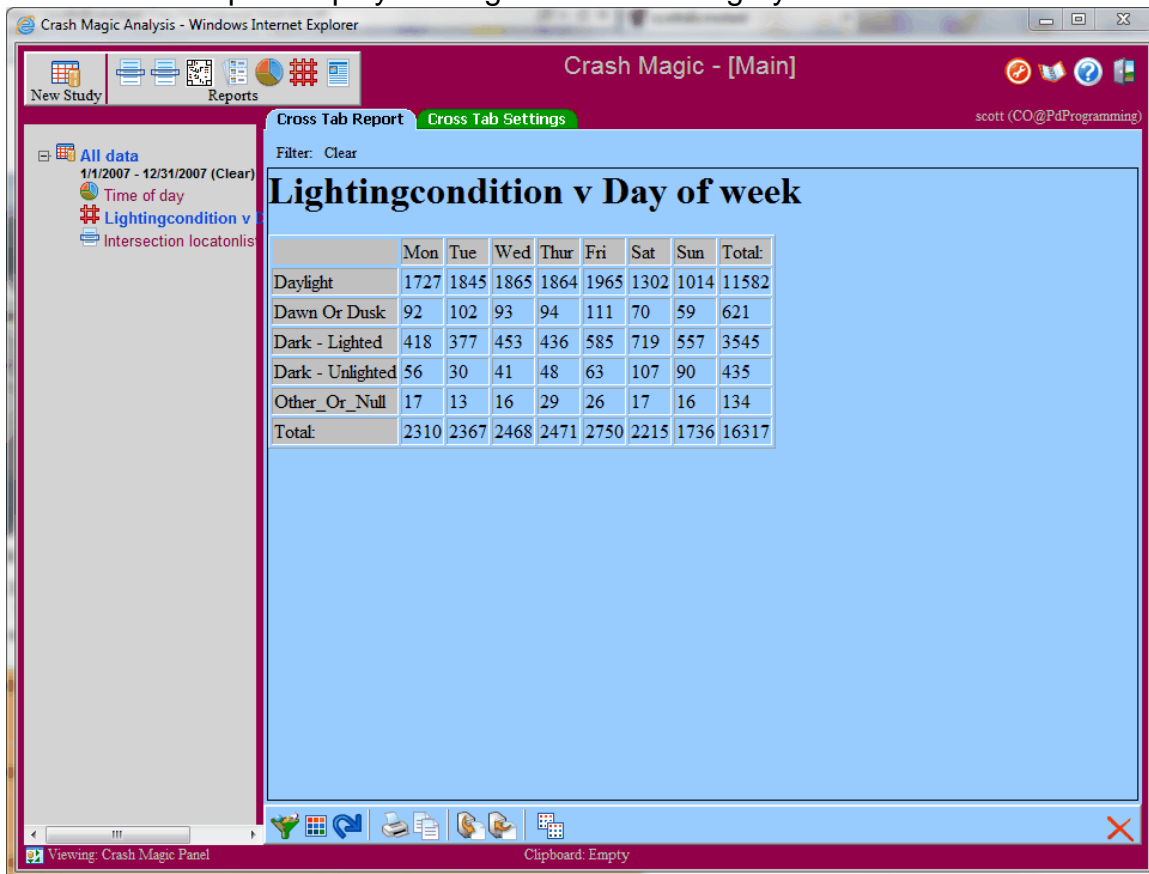
List.XML (PSRO,FormatSpec)	Returns a listing as an XML document.
-----------------------------------	---------------------------------------

Cross Tab

The cross tab panel displays a comparison between two fields using a category set. The cross tab report creates a grid of counts based on the fields of each [category list](#)⁶² in a category set.

Cross Tab report tab

The cross tab report displays a list grid from the category lists selected.



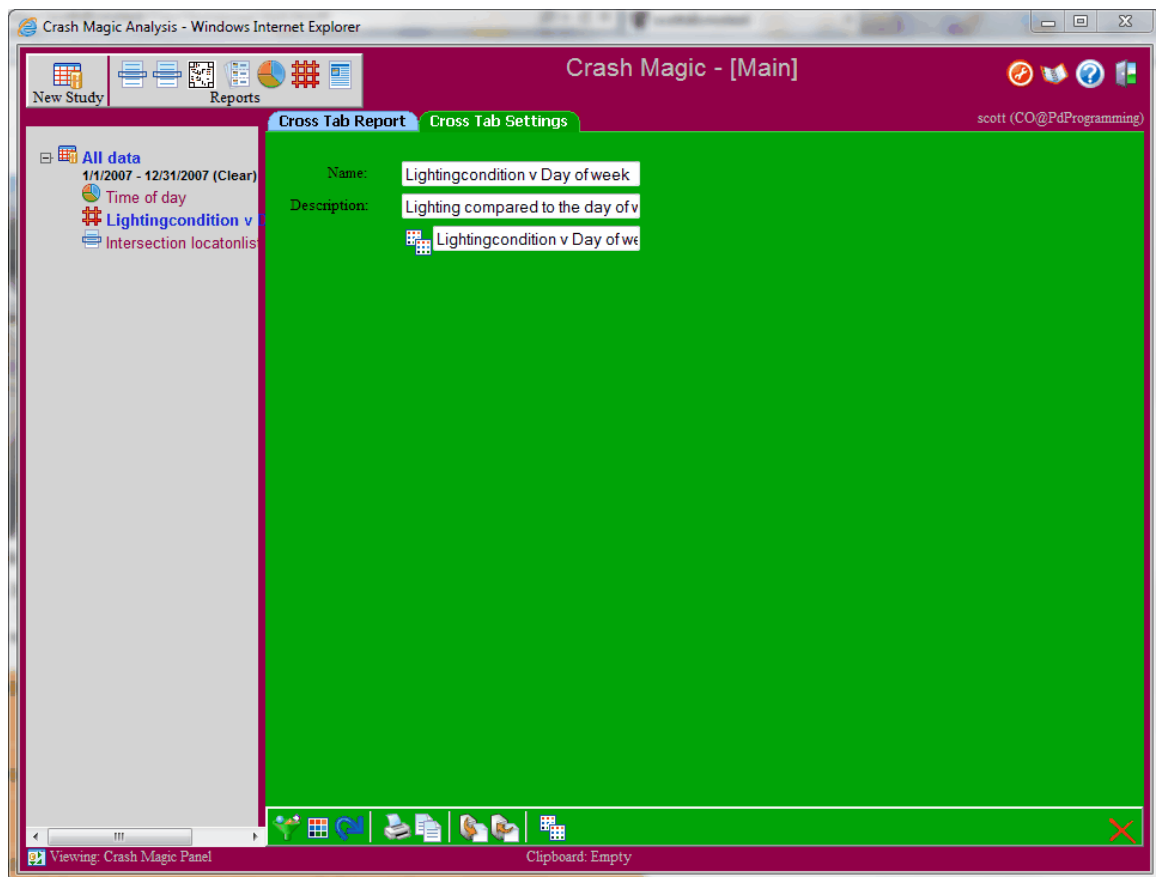
In this example the Lightingcondition category list is compared to the day of week category list. The grid shows the most collisions Mon thru Fri while the lighting condition was Daylight.


Along with the [common function buttons](#)^[66] at the bottom of the panel the following button can be found.

-  - Opens the [category set selection window](#)^[111] for the cross tab report

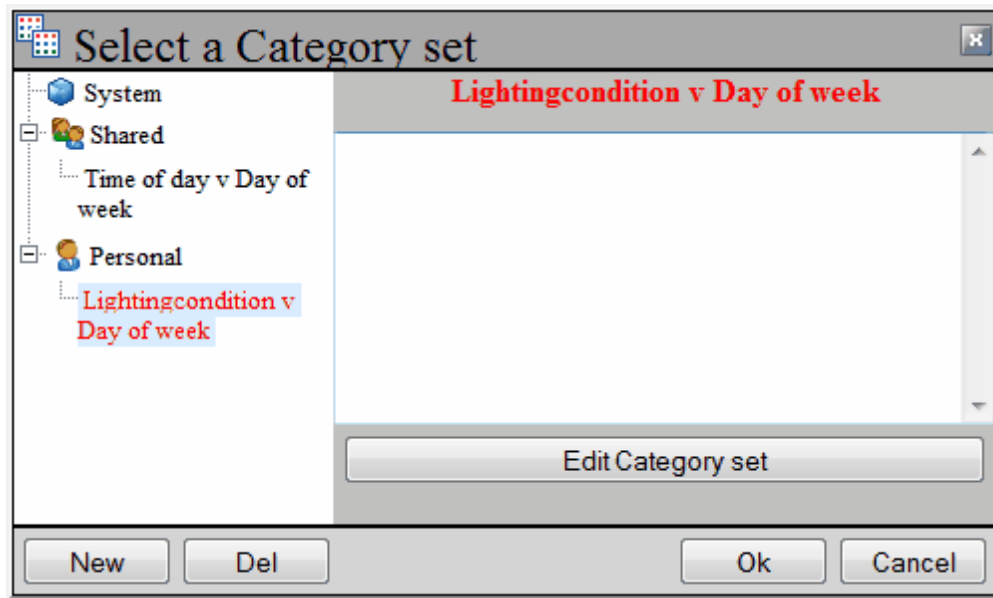
Cross Tab settings tab

The cross tab settings tab controls the display elements of the cross tab report.



- **Name** - Name of the cross tab that will be displayed in the project tree, and as the title of the cross tab report
- **Description** - Description of the cross tab report
-  - Opens the [category set selection window](#)¹¹¹ for the cross tab report

Category set selector



New - Creates a new blank category set named untitled

Del - Deletes the currently selected category set

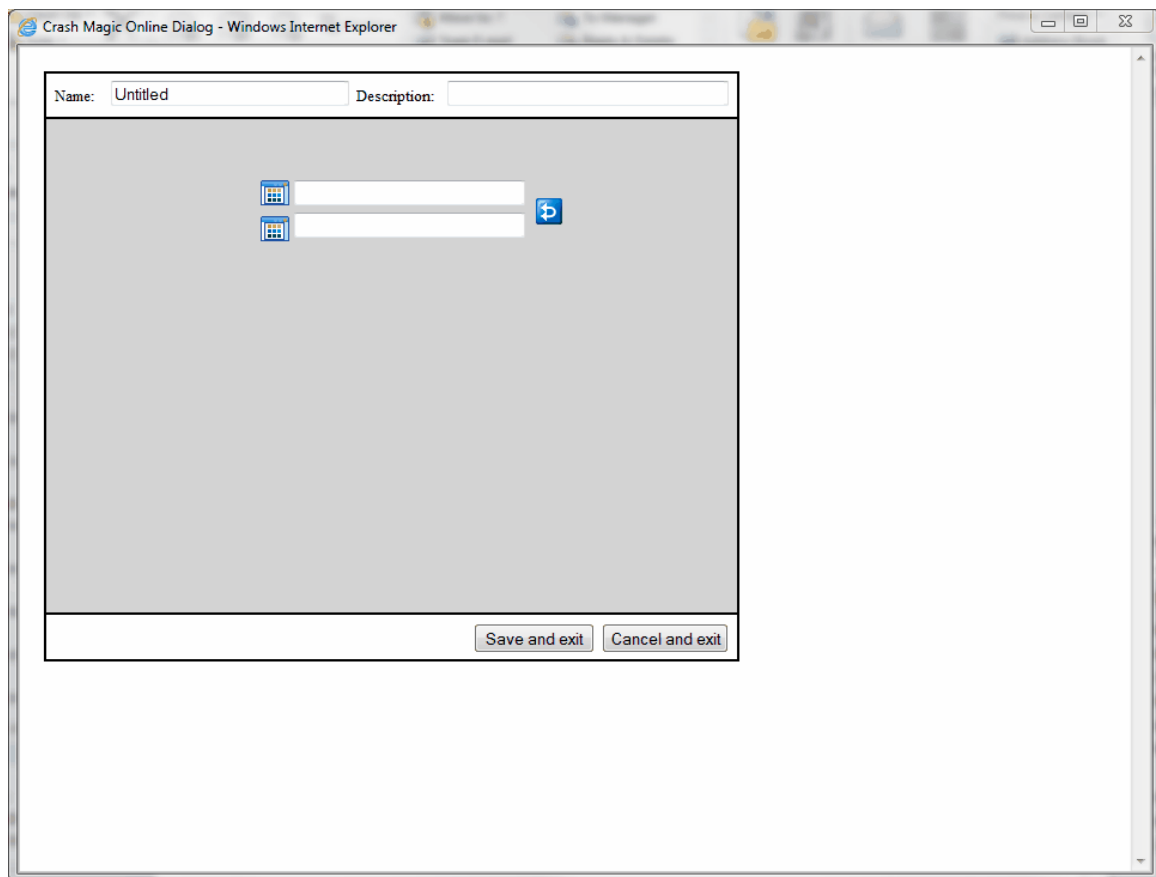
Edit Category Set - Opens the [category set editor](#)^[111] for the selected category set

Ok - Closes the category set selection list box and sets the category set to the category set selected

Cancel - Closes the category set selection list box without selecting the category set

Category set editor


The category set editor allows the user to select two [category list](#)^[62] to compare. The top category selector sets the category list for the Y axis, and the bottom category selector controls the category list for the X axis.



Name - The name of the category set(This is automatically populated with the names of the category lists selected)

Description - Description of the category set

 - Opens the [category list selector](#) ⁶² window

 - Swaps the category list between the two fields

Save and exit - Closes the category set selection list box without selecting the category set

Cancel and exit - Closes the category set editor without making changes to the selection.

High Crash Locations

A location list is used to find locations, and gather statistics across a large dataset. By default the location list shows the intersection or node, along with the crash count and a check box to create a diagram.

High Crash Location report tab

The high crash location list report panel displays a list of locations based on data from the study the report is under.

The screenshot shows the 'Crash Magic - [Main]' window. The 'List report' tab is active, displaying the 'High Crash Location List' for the report run on 5/8/2013 at 2:49:21 PM (GMT-6). The list includes 20 locations with their respective counts. The interface also shows a sidebar with 'All data' and 'Intersection locations' options, and a toolbar at the bottom with various function buttons.

Choose the rank	location	Count
<input type="checkbox"/> 1	S Federal Blvd & W Alameda Ave	62
<input type="checkbox"/> 2	S Santa Fe Dr & W Alameda Ave	61
<input type="checkbox"/> 3	E 6th Ave & N Lincoln St	59
<input type="checkbox"/> 4	E Leetsdale Dr & S Quebec St	59
<input type="checkbox"/> 5	E Leetsdale Dr & S Monaco St	59
<input type="checkbox"/> 6	E Evans Ave & S Monaco Street Pkwy	58
<input type="checkbox"/> 7	E Colfax Ave & N Colorado Blvd	55
<input type="checkbox"/> 8	E Martin Luther King Blvd & N Quebec St	55
<input type="checkbox"/> 9	E 47th Ave & N Peoria St	48
<input type="checkbox"/> 10	N Speer Blvd & W Colfax Ave	47
<input type="checkbox"/> 11	E 39th Ave & N Peoria St	41
<input type="checkbox"/> 12	E 8th Ave & N Colorado Blvd	41
<input type="checkbox"/> 13	E Evans Ave & S University Blvd	40
<input type="checkbox"/> 14	S Santa Fe Dr & W Mississippi Ave	40
<input type="checkbox"/> 15	E Evans Ave & S Colorado Blvd	38
<input type="checkbox"/> 16	E 45th Ave & N Peoria St	36
<input type="checkbox"/> 17	E Alameda Ave & S Colorado Blvd	35
<input type="checkbox"/> 18	E Leetsdale Dr & S Oneida St	35
<input type="checkbox"/> 19	N Kalamath St & W Colfax Ave	34
<input type="checkbox"/> 20	S Federal Blvd & W Evans Ave	34

Along with the [common function buttons](#)^[66] at the bottom of the panel the following button can be found.



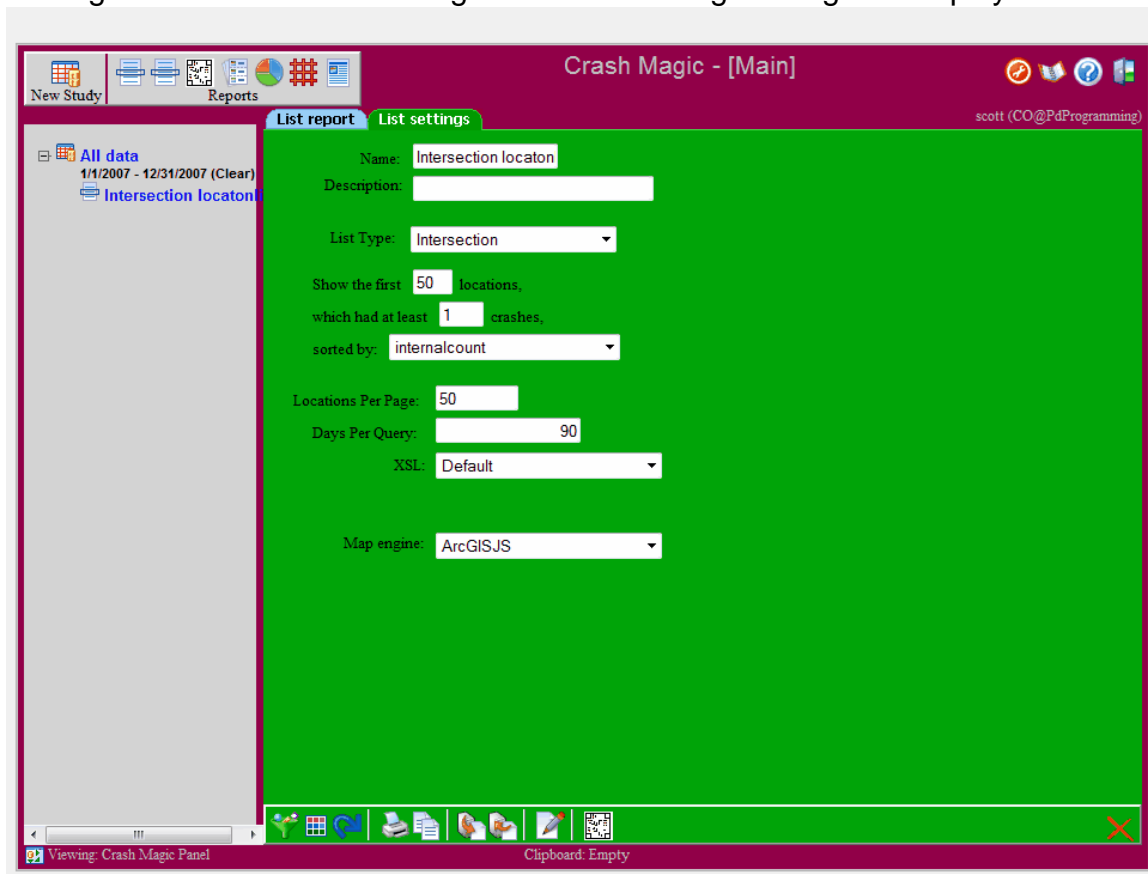
- Opens the [High Crash Location List editor](#)^[115] window.



- Creates a new project with the collision diagrams from the selected locations.

High Crash Location settings tab

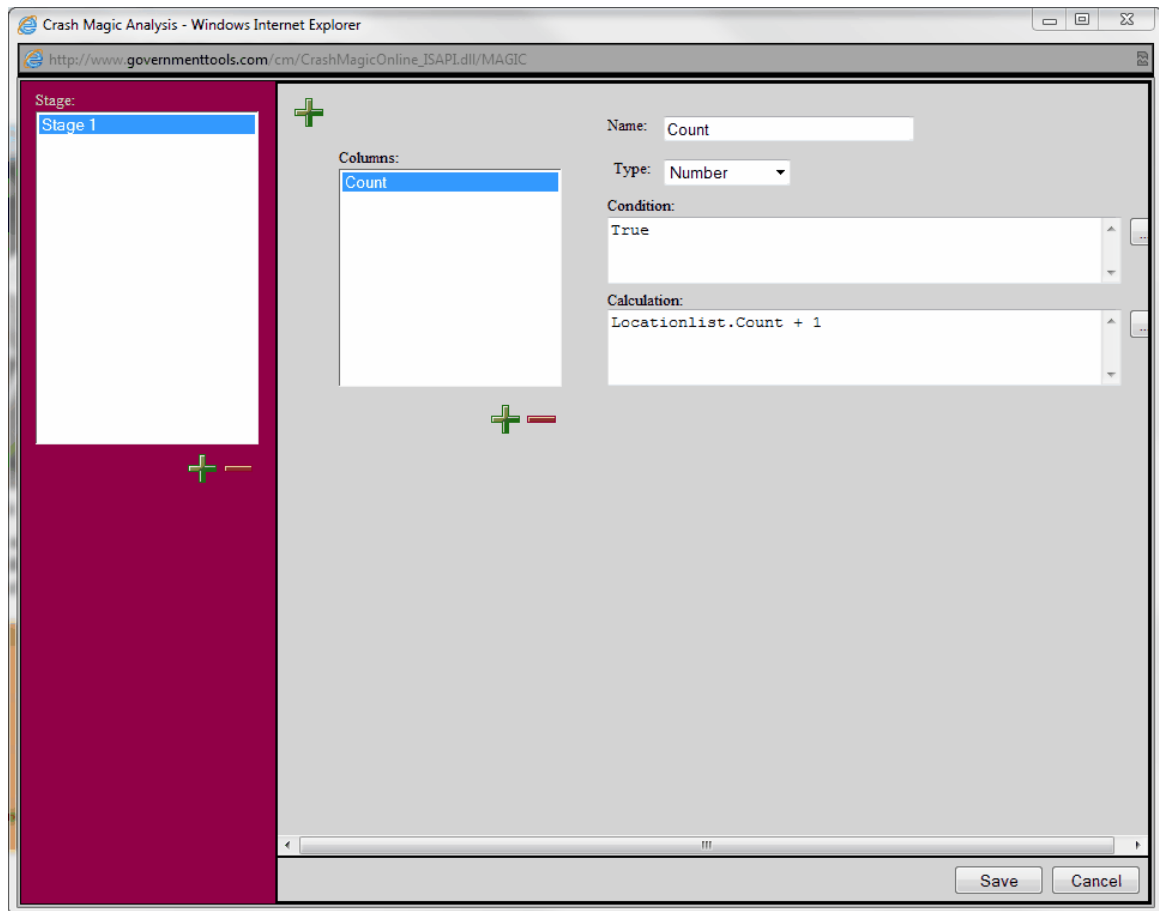
The high crash location list settings tab controls the gathering and display of locations.



- **Name** - Name of the report that will be displayed in the project tree.
- **Description** - Description of the of the list
- **List type** - This is the type of locations being displayed in the list. Node or intersection can be selected.
- **Show the first __ locations** - This limits the number of locations that will be displayed in the report.
- **Which had at least __ crashes** - This limits the report to locations that have a count greater than or equal to the number specified.
- **Sorted by** - Specifies how the list will be sorted.
- **Location Per Page** - Is the number of locations shown on each page of the report.
- **Days Per Query** - High Crash Location reports query the database based on the date range of the study the report is under. Crash Magic accesses the collisions in groups of data based on the number of days specified by this parameter.
- **XSL** - This is the name of the style sheet that will be used to display the list.
- **Map engine** - Specifies the engine that will be used to render a pin map of the locations.

Editor

The High Crash Location list editor allows users to define columns in the High Crash Location list report. The editor is divided into two sections the stage editor and the column editor. The stage editor allows stages to be created for the High Crash Location list. The column editor allows columns to be defined for the High Crash Location list.



- **Stages** - Processing units that columns are created under. Stage_1 is a special stage that works only on the crash records. All other stages work on the location. This means that Locationlist functions will only work on stage_2 or greater. For example when creating a complex column that requires the completed results of a first column. The first column would be created in stage 1 and the second column would need to be created in stage 2 or any later stage there after. For example when creating a simple fatality rate defined as fatal collisions divided by the number of collisions, the count of fatal collisions and the count of collision records would be created under stage 1 as this stage operates on the collision records. The calculation of fatal collisions divided by the collision count must be performed in stage 2 or later as the results from stage 1 are required for the calculation of stage 2.
- **Columns** - Variables displayed as columns in the High Crash Location list.
- **Name** - Name of the variable created to store the calculation and the title of the column displayed. This name cannot contain spaces.

- **Type** - The data type of the variable being created.
- **Condition** - The condition under which a calculation is performed. Crash Magic processes each collision record for a study. When the condition resolves to true for a collision record the calculation is performed.
- **Calculation** - The calculation to be performed. Variables created in the editor must use the Locationlist.<VariableName> designation when used.

Example to create a column with the count of fatal collisions. In this example the column will be named fatal.

1. Click the + in the magenta section to add a Stage 1 for calculation of the fatal column.
2. Click the + sign under the column box to add a new column.
3. Change the name of the column from new to Fatal. This name will be the variable name and the column title.
4. Select a type of Number. The fatal variable will store the number of fatal collisions for a location
5. Enter _ISFATALITY in the condition field. Like a filter the condition field requires a true or false calculation. The _ISFATALITY field is a special field in the P_Default calculated field that returns true if the collision has a fatality. Every time the condition resolves to true the calculation will be performed.
6. In the Calculation field enter Locationlist.Fatal + 1. This calculation tells Crash Magic to increment the Fatal locationlist variable by 1 every time the condition in step 5 is true.
7. Click the save button to see the results.

Example of adding traffic volume data to the High Crash Location list:

1. If there is not already a Stage 1, click the + in the magenta section to add a Stage 1 for calculation of the fatal column.
2. Click the + in the magenta section to add a Stage 2 for calculation of the fatal column(Stage 1 will only operate on the collision data. So we add a Stage 2 to operate on the locations).
3. Click the + sign under the column box to add a new column in Stage 2.
4. Change the name of the column from new to Volume
5. Enter "True" in the condition box. This will tell the High Crash Location list to calculate this value for each location as it is in Stage 2
- 6.

Aggregates have some expressions that are specific to location lists. These expressions can help create advanced calculations including: rates, averages, and totals.

The following aggregate statistics can be applied to any aggregate column (i.e. LocationList.<AggregateName>.<AggregateStatistic>)

Aggregate Statistic	Description
Sum	The sum of an aggregate column.
Mean	The mean of an aggregate column.
Variance	The average variance of an aggregate column. Shows, on average, how widely distributed the aggregate column is.
TotalVariance	The total variance of an aggregate column. Shows how widely distributed the aggregate column is.
Standard Deviation	The standard deviation of an aggregate column.
Skew	The skew (or first moment) of an aggregate column. Shows data symmetry.
Kurtosis	The kurtosis (or second moment) of an aggregate column.
Moment3	The third moment of an aggregate column.
Moment4	The fourth moment of an aggregate column.
MinValue	The minimum value of an aggregate column.
MaxValue	The maximum value of an aggregate column.

Pre Stage Functions

Function	Description
SetActive	SetActive(boolean,[IncludeInactive]) SetActive is used to determine which set of locations are active for the stage. The second parameter is optional and tells the function if it should act on locations that are inactive. By default it only acts on active locations.
Sort	Sort(AggregateName,Desc) Sort is used to sort the location list before a stage.
Top	Top(Number) The top function returns true the location is less than or equal to the number specified. The top function can also act on a percentage when called with a number followed by the "%" symbol.

All aggregates take place inside a stage. Stages are the only way to ensure that aggregates are calculated in a certain order. Each stage consists of two parts, the pre stage, and the aggregates.

The pre-stage processes are used to perform actions on all the rows before the stage takes place (like sorting, or enabling/disabling a row).

Example of a pre stage section:

```

<PreStage>
  <Process>
    <Condition>
      <Line>True</Line>
    </Condition>
    <Command>
      <Line>LocationList.SetActive(False, True)</Line>
    </Command>
    <ActOnEachLocation>True</ActOnEachLocation>
  </Process>
</PreStage>

```

XML Tag	Description
PreStage	There is one PreStage tag per Stage section. The PreStage tag holds all of the Process tags.
Process	The Process tag is used to group each process together. There is one process tag for each process.
ActOnEachLocation	The ActOnEachLocation tag accepts either True or False. If set to true, the process will run for each row of the location list. If false, the process will only run once for the entire list.
Condition	The Condition is a boolean expression that determines if the command runs.
Command	The Command is an expression that acts on the location list. These commands are specific to the location list.

The AggregateCol is a column that will appear in the location list. It will also be available in future stages, accessible by its name.

Example of a pre stage section:

```

<AggregateCol>
  <Name>Count</Name>
  <Visible>True</Visible>
  <Condition>
    <Line>True</Line>
  </Condition>
  <DisableEscaping>False</DisableEscaping>
  <Calculation>
    <Expression>
      <Line>LocationList.FirstCount</Line>
    </Expression>
    <Type>Number</Type>
  </Calculation>
  <DisplaySettings>
    <Format>
      <Line>AsStringF(This, "%g")</Line>
    </Format>
  </DisplaySettings>
  <Summaries>
    <Summary>
      <Name>Avg</Name>
    </Summary>
  </Summaries>
</AggregateCol>

```

```

        <Calculation>
            <Expression>
                <Line>LocationList.Count.Mean</Line>
            </Expression>
            <Type>Number</Type>
        </Calculation>
        <DisplaySettings>
            <Format>
                <Line/>
            </Format>
        </DisplaySettings>
    </Summary>
</Summaries>
</AggregateCol>

```

XML Tag	Description
AggregateCol	The AggregateCol tag holds the information for each aggregate column. There is an AggregateCol for each column.
Name	The name of the aggregate column. This will appear as the header for the column. This will also be used in expressions (accessible by using LocationList.<Name>).
Visable	If true, the column will display in the location list. If false, the column will not appear in the location list (but will still be accessible in future stages).
Condition	The Condition tag is a boolean expression that determines if the calculation is evaluated or not.
Calculation	The Calculation tag holds the Expression and Type. There is one Calculation tag per AggregateCol tag.
Expresion	The Expression determines the value of the aggregate column.
Type	The type that the expression will result in.
DisplaySettings	Holds information for formatting and displaying this aggregate column.
Format	The expression that defines the final output of the aggregate column. The keyword "This" can be used to refer to the current AggregateCol.

Summaries appear at the bottom of aggregate columns. They are used to display information that is relative to the entire column (like averages or totals).

```

<Summaries>
    <Summary>
        <Name>Avg</Name>
        <Calculation>
            <Expression>
                <Line>LocationList.Count.Mean</Line>
            </Expression>
            <Type>Number</Type>
        </Calculation>
        <DisplaySettings>
            <Format>
                <Line/>
            </Format>
        </DisplaySettings>
    </Summary>
</Summaries>

```

```

</DisplaySettings>
</Summary>
</Summaries>

```

XML Tag	Description
Summary	The Summary tag holds information to display a summary.
Name	The name of the summary. Summaries of the same name will be displayed in the same row.
Calculation	The Calculation tag holds the Expression and Type. There is one Calculation tag per AggregateCol.
Expression	The Expression determines the value of the summary.
Type	The type that the expression will result in.
Display Settings	Holds information for formatting and displaying this aggregate column.
Format	The expression that defines the final output of the aggregate column. The keyword "This" can be used to refer to the current AggregateCol.

Layout Elements

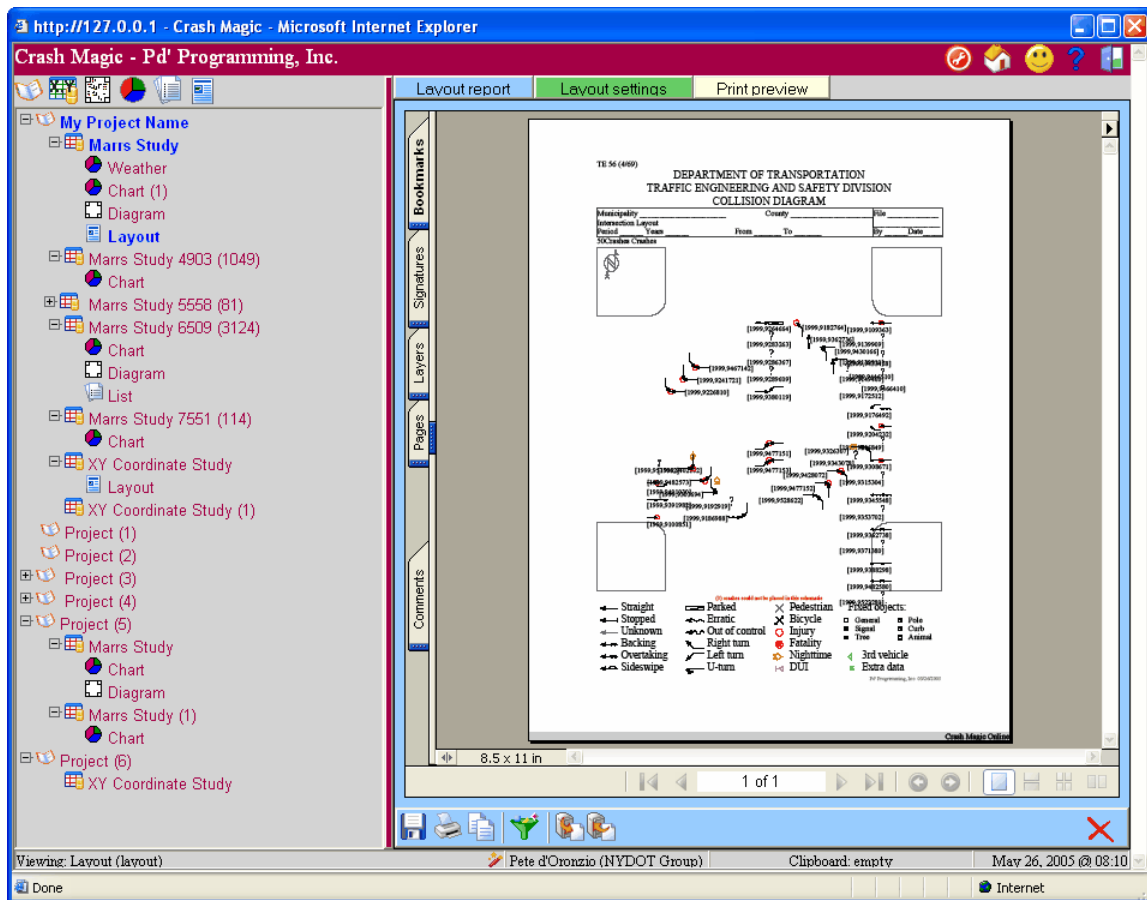
LocationList.FO (PSRO,FormatSpec)	Returns a location list ready to be placed inside of a FO document.	
	FormatSpec:	
	StartIndex	The item number (starting at 0) to start outputting results at. <i>Default: 0</i>
	NumberOfResults	The number of results to output. <i>Default: 50</i>
	XSLName	The name of the xsl-locationlist attribute to output. <i>Default: Report defined</i>
LocationList.XML (PSRO,FormatSpec)	DaysPerQuery	The number of days to query at a time when processing the report. <i>Default: Report defined</i>
	Returns a location list in XML format.	
	FormatSpec:	
	StartIndex	The item number (starting at 0) to start outputting results at. <i>Default: 0</i>

NumberOf Results	The number of results to output. <i>Default: 50</i>
DaysPerQuery	The number of days to query at a time when processing the report. <i>Default: Report defined</i>

Layouts

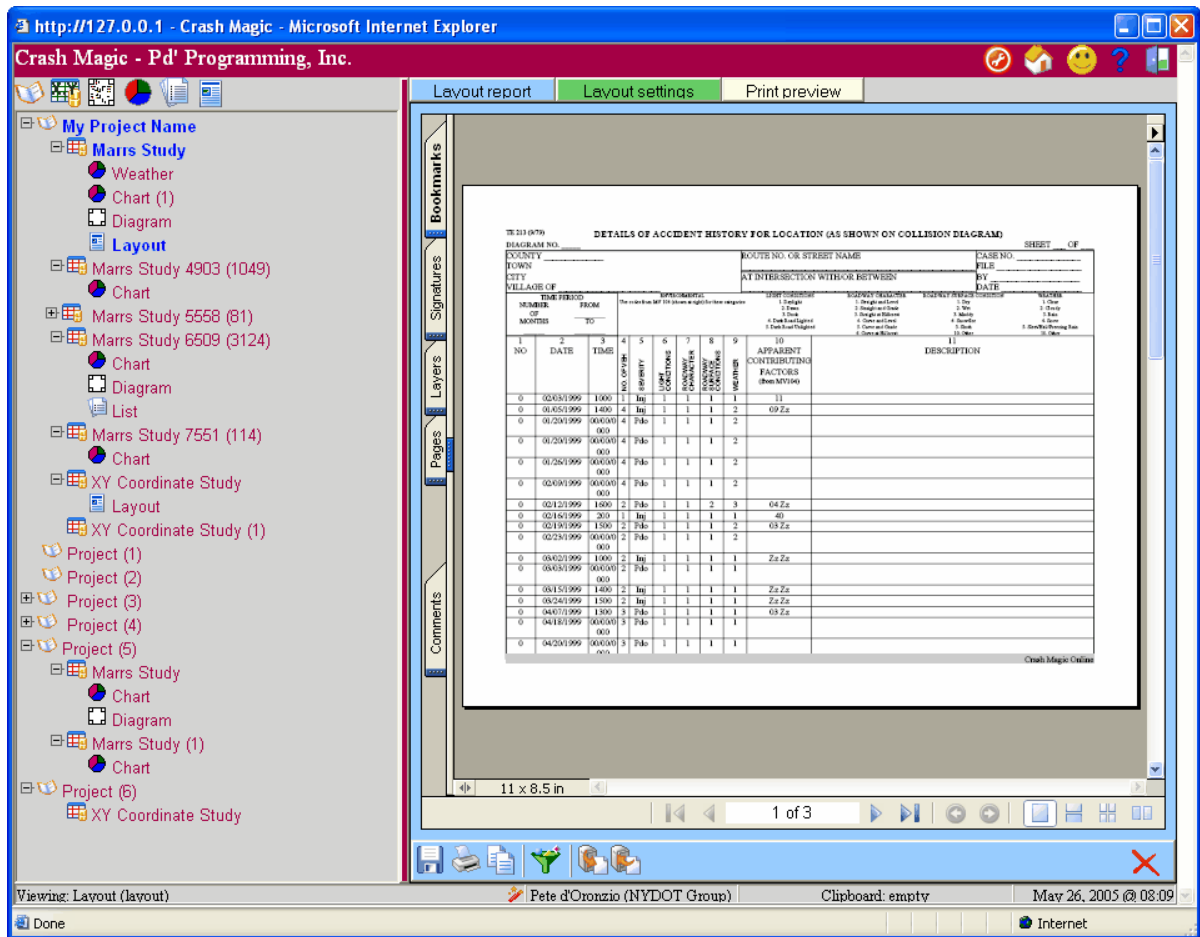
The layout panel is used for creating print (pdf) layouts. A layout describes where various report components will be displayed on a printed page. Layouts can be general in nature so that it is possible to present any report in the main part of the page, or a layout can be specific to a particular data set, agency specification, etc.

Here is an example of a diagram layout that was created to emulate a printed report that was the standard for a specific DOT. Note that the diagram is printed without a header, but with a key. The table above is populated partially by user input, and partially from program information.



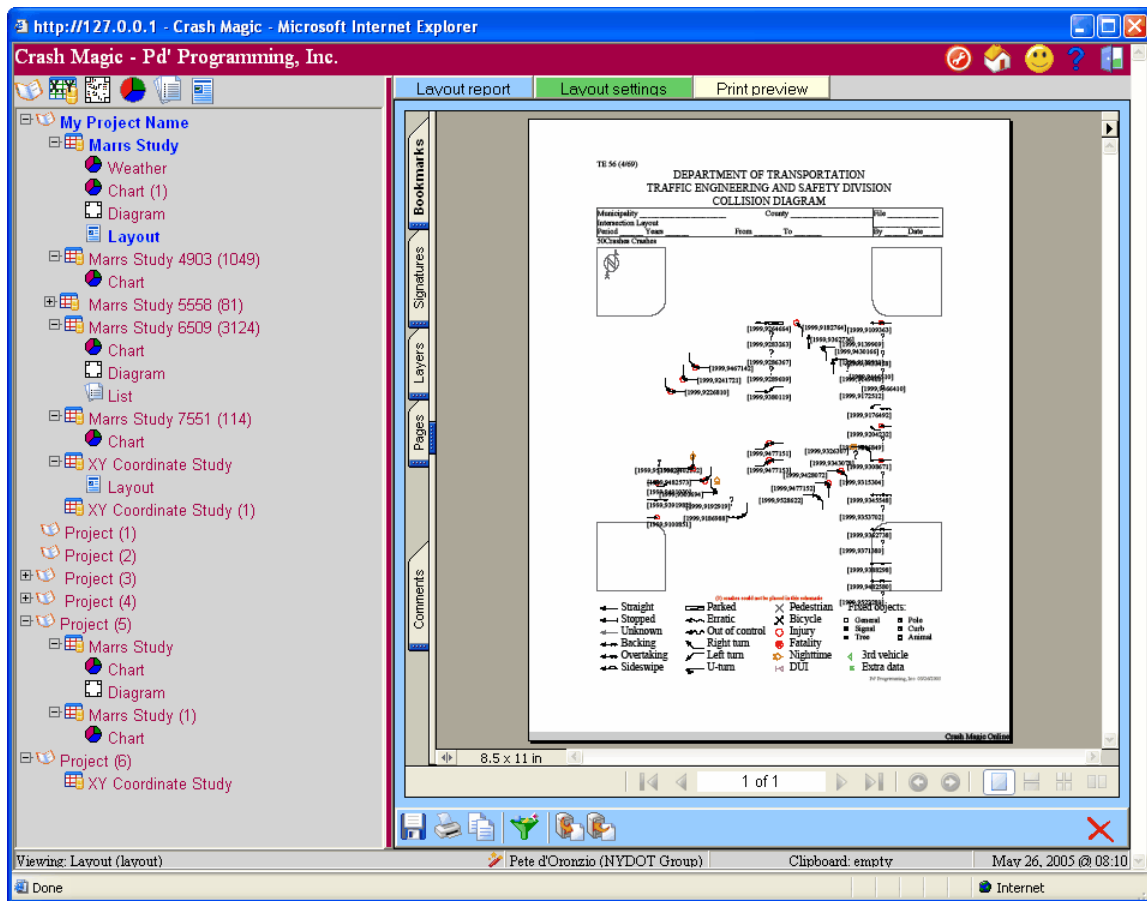
Here is another layout example. This layout is essentially a listing, but the field values are not taken from the lookup tables, and the report header is much more complicated

than a simple tabular listing of field names. Again, this custom layout was created to emulate an existing form from a DOT.



Layout report tab

The Layout report tab displays the layout after rendering as pdf.

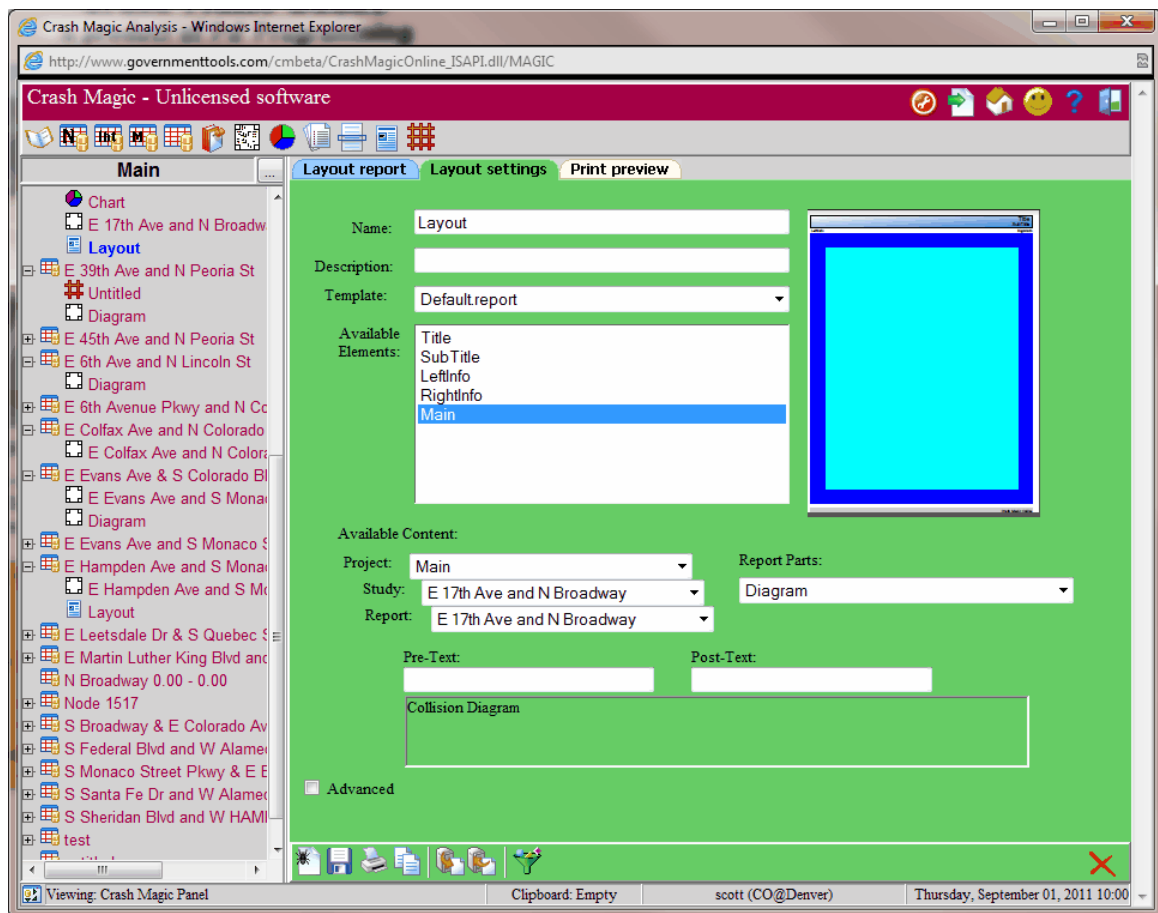


Layout settings tab

The layout settings tab provides an interface for specifying the content that should be inserted into a fo template to create a usable layout.

The FO template describes the page layout. XSL-FO is a W3C XML specification that describes a printed page. Crash Magic uses a FO template containing special tags where Crash Magic report elements will be inserted. (i.e. a diagram, chart, number, text, etc.) After selecting a FO template, the list of report elements will be updated below the FO template name. Each of the report elements must have a valid expression (below) that generates appropriate output for the element.

Once the layout settings have been described, the layout may be generated. Click the report tab to view the results. Any errors will be shown inline on the pdf page.



In this image the Default.report template selected contains five report elements that can be filled from the available content selection. The Main element is currently being filled with a diagram from the E17th Ave and N Broadway report under the E17th Ave and N Broadway study under the Main project.

- **Name** - This is the name of the layout report that will show in the project tree.
- **Description** - This is the description of the layout report
- **Template** - This list box contains all the different styles of reports that Crash Magic knows how to output. When selected, a small preview will appear to the right.
- **Available Elements** - These are the names of every element that can contain content in the report based on the Template selected. Selecting an element will cause it to change cyan on right preview. Once an element is selected, it can be filled using the "Available Content" boxes to select items from the project tree.
- **Available Content** - The Report Parts box allows user to select content for the selected element from the project, study and report selected. Changing any of the left three boxes will cause the Report Parts box to fill with new content.
 - Project - Allows users to select from various [projects](#)^[90] they have created.
 - Study - Allows users to select [studies](#)^[136] under the above project selected (Selecting None will restrict content to only report parts that can be provided by the project selected).
 - Report - Allows user to select [reports](#)^[94] under the above studies selected.

- Report parts - Allows users to select parts for the current element based on the project, study and report selected.
- **Pre-Text/Post-Text** - These fields allow you to put a word or phrase before or after the content. This is very useful when selecting content that is a number and needs some context (from crash count you might want to add the word "Crashes" to post-text).
- **Advanced** - Advanced allows the user to write an expression to describe the content. This is often only used by very experienced users of Crash Magic.

Layout elements

The expressions that are used to create layout elements include all standard filter functions, as well as a number of functions that are specifically for layouts. These layout specific functions provide access to any projects, studies or reports in the project tree. Also available are any templates that have been saved to the server.

The following functions and variables are available:

Most of the functions below take a parameter called PSRO.

PSRO is a text string that specifies the desired Project, Study, Report and ObjType. The format of this string is

Project:<ProjectName>;Study:<StudyName>;Report:<ReportName>;ObjType:<ObjType>. Here is an example of a PSRO string: "Project:Project one;Study:Test study 1;Report:My diagram;ObjType:diagram"

Because the system always knows the names of the Project, Study and Report, these values can be filled in automatically. If any of these parameters are missing, the values for the currently selected objects will be used. For example, to create an expression that always selects the current project, current study, and the diagram called "rpt diagram", you would use the following PSRO string:

"Report:rpt diagram;ObjType:diagram".

Finally, this string can be left completely blank to specify the *current* project, study and report. This is how to create a layout for a single, current report. A typical means of creating a re-usable report is to create it with a specific PSRO until it is working properly. Once it is complete, change the PSRO to a blank string so that the report will use the currently selected report for its content.

At the time of creation of this manual, the only available project ObjType value is "project". ObjType for studies can be "marrs", "intersection", "xystudy", "milepost", "daterange". ObjType for reports are "diagram", "chart", "list", "layout", "map".

Most of the functions also take a parameter called FormatSpec.

FormatSpec is used to specify formatting options for the specified content. For example, it may contain font information, size, or content-specific information. Most content has reasonable default values if the FormatSpec doesn't include any values. The format spec is also semi-colon delimited. Here is an example of a FormatSpec for a collision diagram object: "width:7.5in;ShowKey:True".

CurrentProject**CurrentStudy****CurrentReport****CurrentObjType**

Project functions:

Project.Name.ASCII

Project.Name.HTML

Project.Name.SVG

Project.Name.FO

Project.Description.ASCII

Project.Description.HTML

Project.Description.SVG

Project.Description.FO

Project.DB.XML

Study functions:

Study.Name.ASCII

Study.Name.HTML

Study.Name.SVG

Study.Name.FO

Study.Description.ASCII

Study.Description.HTML

Study.Description.SVG

Study.Description.FO

Study.DB.XML**CurrentProject**

Returns a string that specifies the name of the currently selected project.

CurrentStudy

Returns a string that specifies the name of the currently selected study.

CurrentReport

Returns a string that specifies the name of the currently selected report.

CurrentObjType

Returns a string that specifies the objtype of the currently selected project, study or report. (i.e. chart, diagram, list, layout)

Project.Name.ASCII(PSRO,FormatSpec)

Returns the name of the specified project in the format requested.

Project.Description.ASCII(PSRO,FormatSpec)

Returns the description of the specified project in the format requested.

Project.DB.XML(PSRO,FormatSpec)

Returns the raw xml document stored with the specified project. Using this content is a very advanced subject which requires an understanding of the Crash Magic internals as well as XML processing.

Study.Name.ASCII(PSRO,FormatSpec)

Returns the name of the specified study in the format requested.

ObjTyp for studies can be "marrs", "intersection", "xystudy", "milepost", "daterange"

Study.Description.ASCII(PSRO,FormatSpec)

Returns the description of the specified study in the format requested.

Study.DB.XML(PSRO,FormatSpec)

	Returns the raw xml document stored with the specified study. Using this content is a very advanced subject which requires an understanding of the Crash Magic internals as well as XML processing.
Study.FilterName.ASCII	Study.FilterName.ASCII (PSRO,FormatSpec) Returns a string containing the name of the filter associated with the specified study.
Study.RecordCount.ASCII	Study.RecordCount.ASCII (PSRO,FormatSpec) Returns a string containing the number of records in the specified study. This is the number of records returned from the SQL query. It does not take into account the filter, if any, associated with the study.
Study.FilteredRecordCount.ASCII	Study.FilteredRecordCount.ASCII (PSRO,FormatSpec) Returns a string containing the number of records in the specified study <i>after filtering</i> . This is the number of records returned from the SQL query less the records removed as a result of the filter, if any, associated with the study.
Study.QueryName.ASCII Study.QueryName.HTML Study.QueryName.SVG Study.QueryName.FO	Study.QueryName.ASCII (PSRO,FormatSpec) Returns a string containing the name of the query associated with the specified study.
Study.List.XML	Study.List.XML (PSRO,FormatSpec) Returns an XML document containing the requested fields. Because this is a Study function, all of the fields from the specified study are available. The FormatSpec should include a variable called "ListTemplate". This variable must point to a template containing a list template containing the desired fields for this report.
Report functions (common): <report>.Name.ASCII <report>.Name.HTML <report>.Name.SVG <report>.Name.FO	<report>.Name.ASCII (PSRO,FormatSpec) Returns the name of the specified report in the format requested. <report> can currently be: Diagram,Chart,List or Layout.
<report>.Description.ASCII <report>.Description.HTML <report>.Description.SVG <report>.Description.FO	<report>.Description.ASCII (PSRO,FormatSpec) Returns the description of the specified report in the format requested. <report> can currently be: Diagram,Chart,List or Layout.

<report>.List.XML

<report>.DB.XML(PSRO,FormatSpec)

Returns the raw xml document stored with the specified report. Using this content is a very advanced subject which requires an understanding of the Crash Magic internals as well as XML processing.

PSRAttr functions:

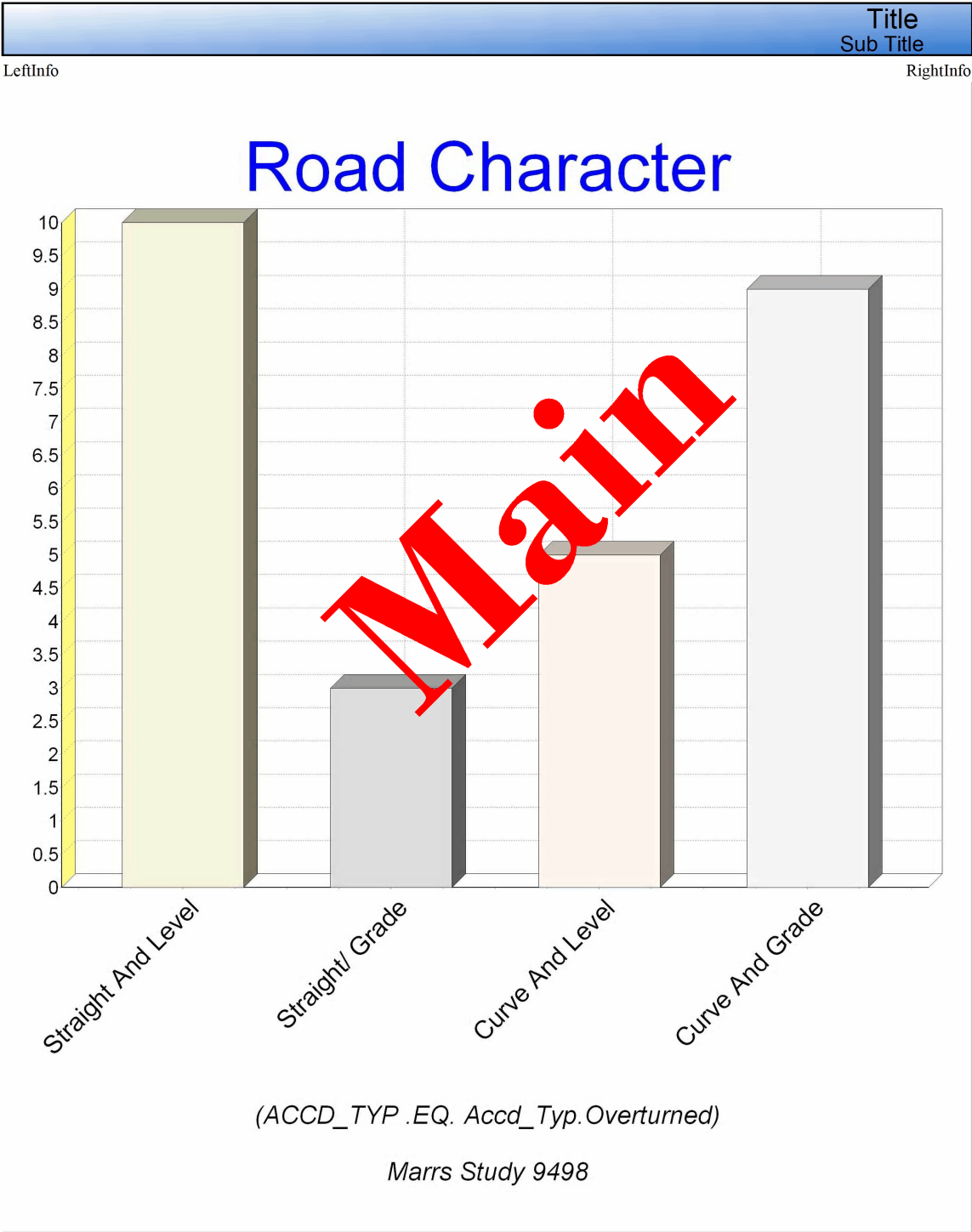
Template.XSL

Template.XSL(PSRAttrName, PSRAttrObjectType, OutputType)

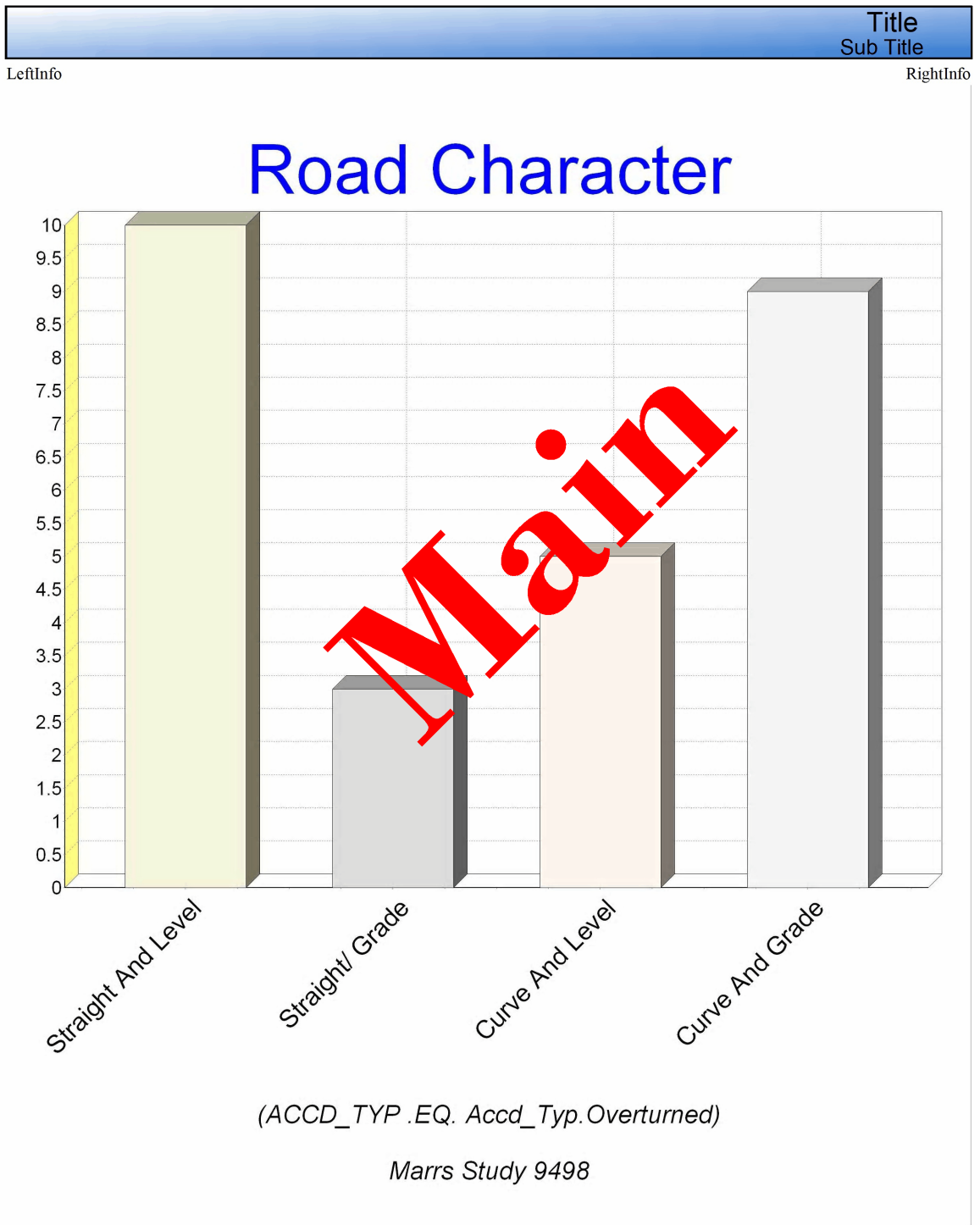
Returns the XSL document inside of a PSRAttr. For backwards compatibility with current layouts, if output type is not specified, and there is only one XSL document inside of the PSRAttr, it will select that document.

FO Templates

There are a variety of FO templates included with Crash Magic, and additional ones may be created and added to the system. This section documents the templates that ship with Crash Magic by default.



Crash Magic Online



TE 213 (9/79)

DETAILS OF ACCIDENT HISTORY FOR LOCATION (AS SHOWN ON COLLISION DIAGRAM)

SHEET OF

DIAGRAM NO. _____

COUNTY _____
TOWN _____
CITY _____
VILLAGE OF _____

ROUTE NO. OR STREET NAME _____
AT INTERSECTION WITH/OR BETWEEN _____

CASE NO. _____
FILE _____
BY _____
DATE _____

TIME PERIOD
NUMBER OF MONTHS FROM _____ TO _____

ENVIRONMENTAL
Use codes from MV 104 (shown at right) for these categories

LIGHT CONDITIONS
1. Daylight
2. Dawn
3. Dusk
4. Dark Road Lighted
5. Dark Road Unlighted

ROADWAY CHARACTER
1. Straight and Level
2. Straight and Grade
3. Straight at Hillcrest
4. Curve and Level
5. Curve and Grade
6. Curve at Hillcrest

ROADWAY SURFACE CONDITION
1. Dry
2. Wet
3. Muddy
4. Snow/Ice
5. Slush
10. Other

WEATHER
1. Clear
2. Cloudy
3. Rain
4. Snow
5. Sleet/Hail/Freezing Rain
10. Other

Road Character

Road Character	Count
Straight And Level	10
Straight Grade	3
Curve And Level	5
Curve And Grade	9

(ACCD_TYP, EQ, Accd_Typ, Overturned)

Mars Study 9498

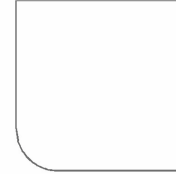
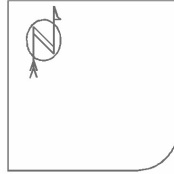
TE 56 (4/69)

DEPARTMENT OF TRANSPORTATION
TRAFFIC ENGINEERING AND SAFETY DIVISION
COLLISION DIAGRAM

Municipality _____	County _____	File _____
Intersection Title _____		By _____
Period _____	Years _____ From _____ To _____	Date _____

LeftInfo Crashes

RightInfo



[1999.0, 0.0, 9621.45]

[1999.0, 0.0, 95300.99]

[1999.0, 0.0, 94440.99]

[1999.0, 0.0, 94250.9]

[1999.0, 0.0, 9101.227]

[2001.0, 0.0, 15492.7]

[2000.0, 0.0, 94600.95]

[2000.0, 0.0, 93171.75]

[2000.0, 0.0, 92281.69]

[2000.0, 0.0, 1035.1]

[2000.0, 0.0, 15409]

[2000.0, 0.0, 9581.7]

[2000.0, 0.0, 92039.97]

[2000.0, 0.0, 92230.0]

[2000.0, 0.0, 92389.03]

[2000.0, 0.0, 92698.54]

[2000.0, 0.0, 93155.1]

[2000.0, 0.0, 94308.2]

[2000.0, 0.0, 94399.7]

[2000.0, 0.0, 95725.50]

[2001.0, 0.0, 15492.5]

[1999.0, 0.0, 97711.3]

[1999.0, 0.0, 92559.7]

[1999.0, 0.0, 92800.7]

Main

(0) crashes could not be placed in this schematic

← Straight
 ← Stopped
 ← Unknown
 ↔ Backing
 ↔ Overtaking
 ↔ Sideswipe

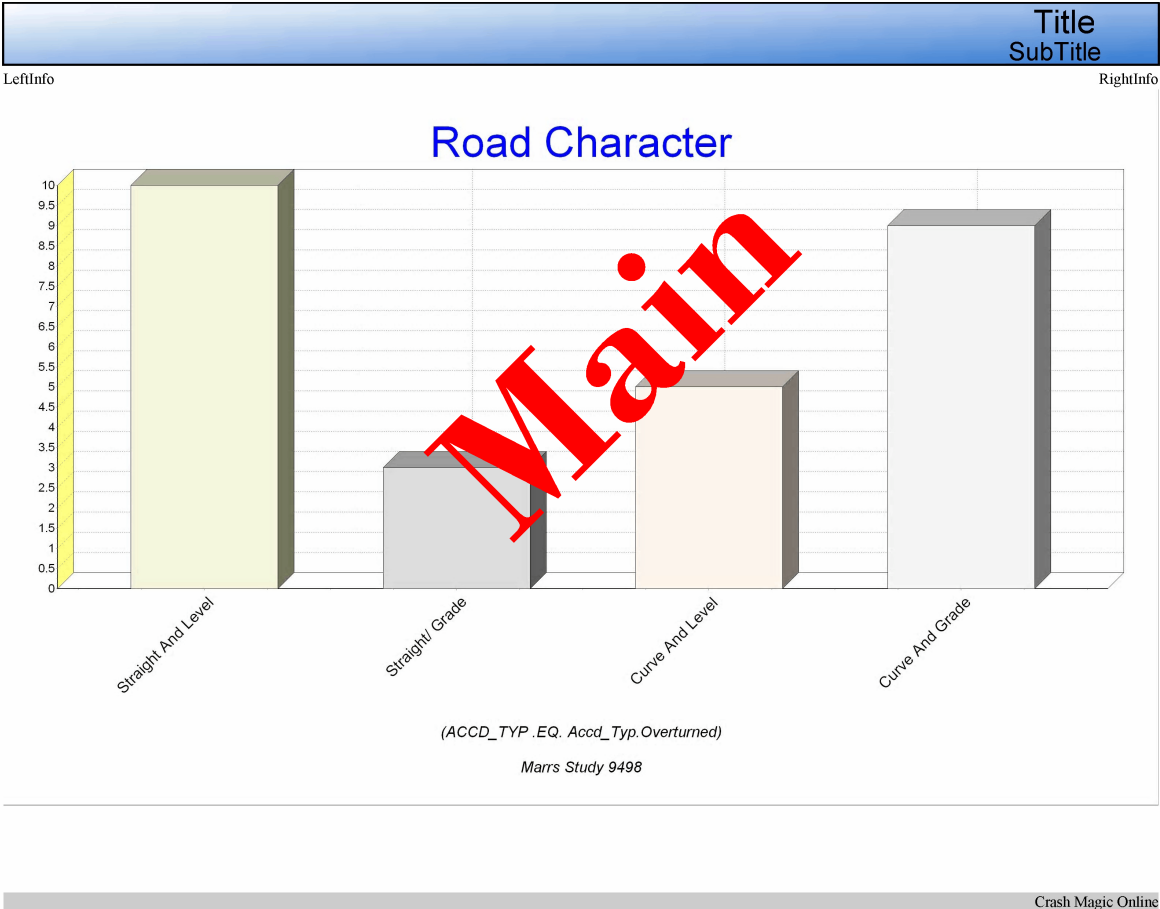
▬ Parked
 ~ Erratic
 ~ Out of control
 ↗ Right turn
 ↖ Left turn
 ↻ U-turn

× Pedestrian
 × Bicycle
 ○ Injury
 ⊙ Fatality
 ⚡ Nighttime
 ⚠ DUI

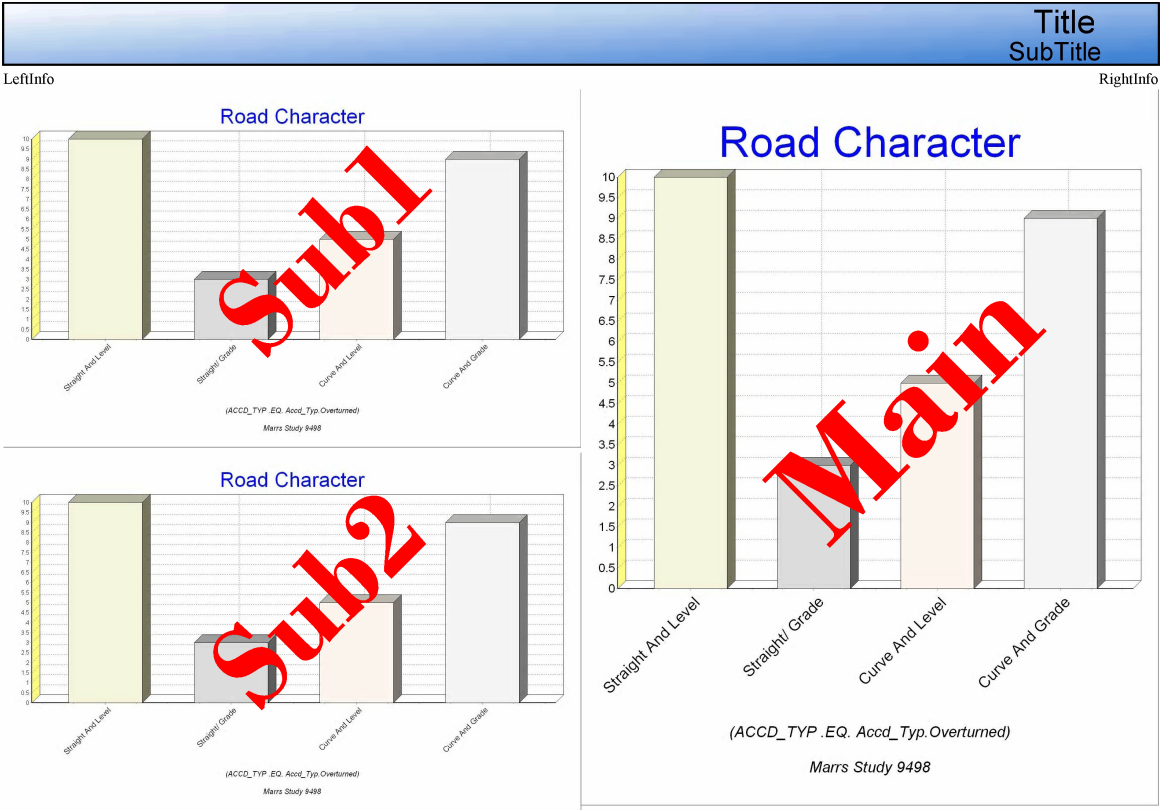
Fixed objects:

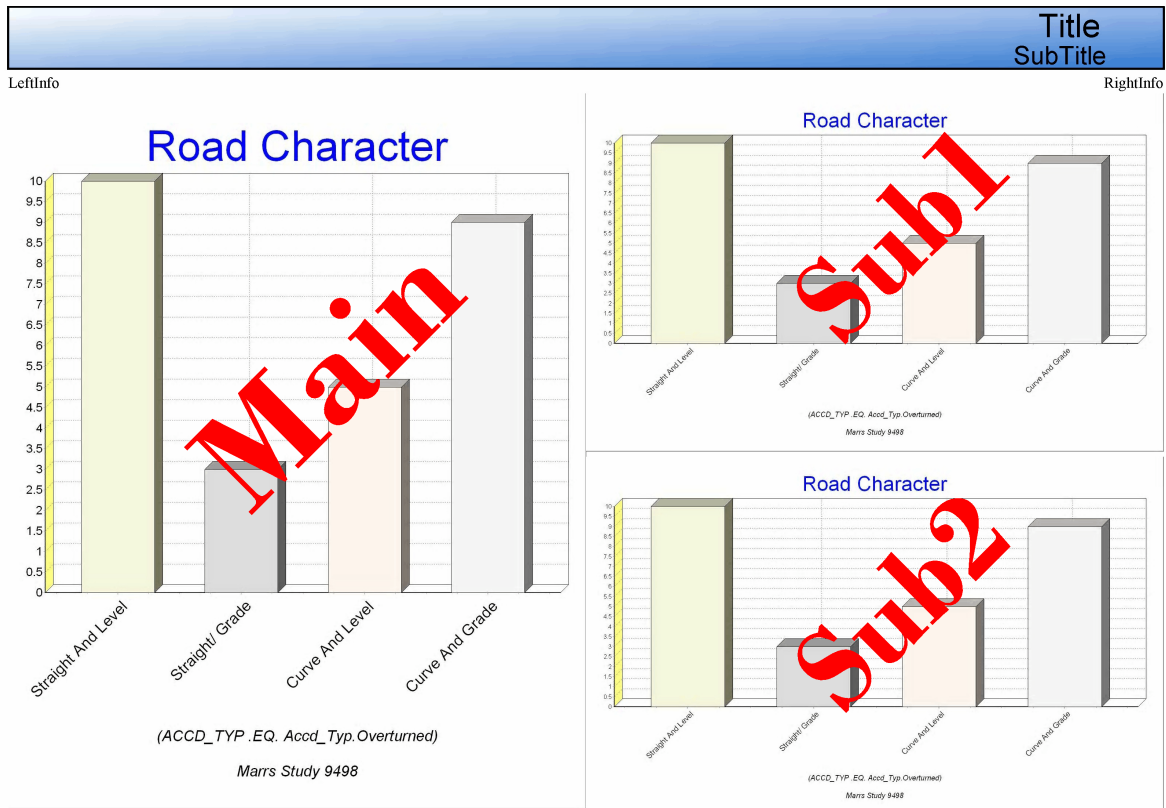
☐ General ☐ Pole
☐ Signal ☐ Curb
☐ Tree ☐ Animal
 ◁ 3rd vehicle
 * Extra data

Pd' Programming, Inc. 01/25/2006









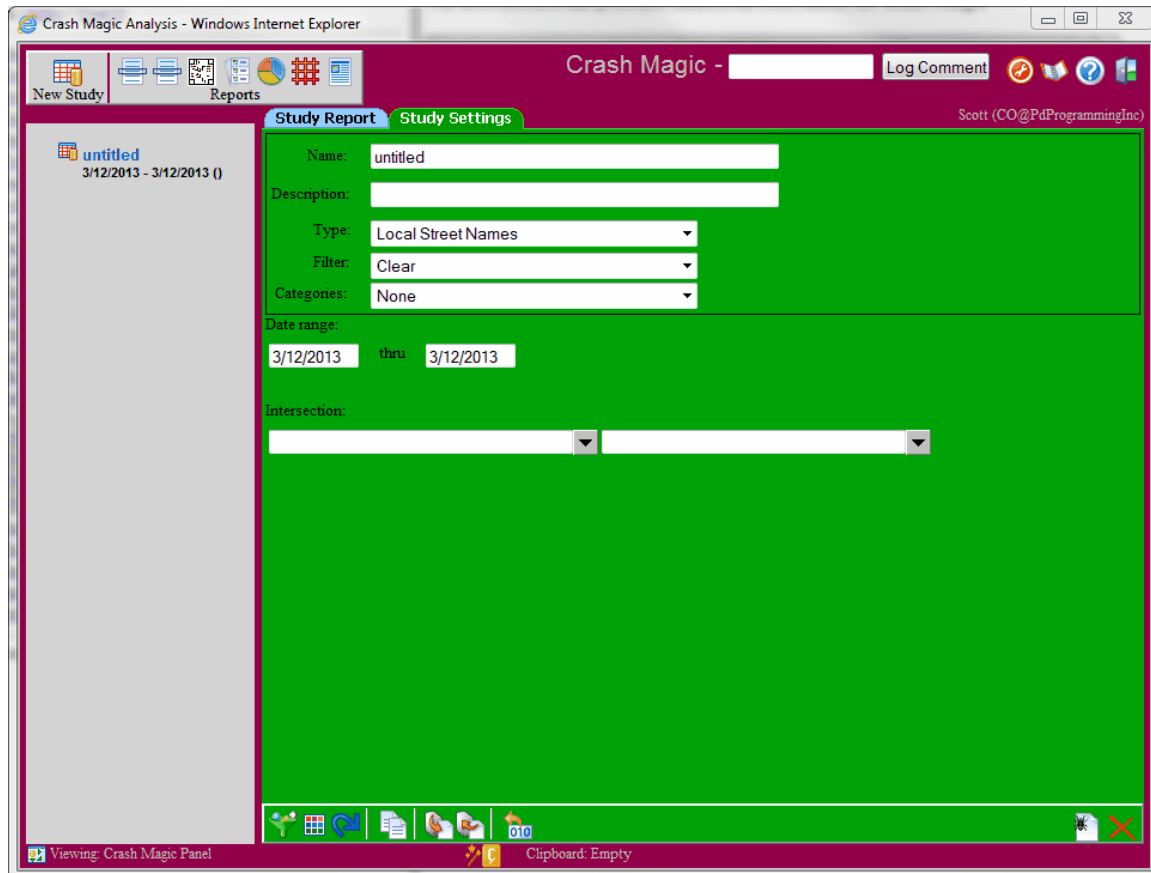
Crash Magic Online

4.12 Studies

The Crash Magic study is the container for crash data. All reports in the system must reference a study containing a group of crashes for analysis. There are studies for gathering intersection crashes, milepost crashes, GIS located crashes, etc. The various study types are described in the [study types](#) ¹⁴⁰ section.

Functionally, the study only contains connection, query and parameter information to gather the crash data. This distinction means that a report generated Monday will contain different content than a report generated Tuesday if the data has changed between the generation of the two reports. This is because the report is generated from scratch each time.

Study settings tab



An intersection study is shown in this image. The fields "Start date", "End date", "Primary street" and "Cross street" are specific to the intersection study definition. The specific fields displayed may be different depending on the study and query used.

The study settings panel provides controls for setting the name, description and study-specific attributes. Each study has at least one query, parameters for that query, a filter and related documents.

The following is a list of the common study controls found in the panel:

- **Name** - This is the name of the study. The name is displayed in the project tree.
- **Descriptions** - This field allows users to enter a short description of the the study.
- **Type** - This is a selectable list of available [study types](#)^[140]. The types of studies available depend on the [study definitions](#)^[280] created for the data.
- **Filter** - This box shows any [filters](#)^[76] that are currently applied to the study. Users can also edit the filter from this box.
- **Categories** - This drop down menu allows a [category list](#)^[62] to be selected. The category selected in this drop down menu will highlight diagrams and listing reports created under the study.









Study Fields

Specific study fields are displayed for the appropriate study type selected. The study types displayed are specific to the study definitions created for collision data in use. The following are a list of possible study fields shown:

- Block address start
- Block address end
- Date range start date
- Date range end date
- Primary street
- Intersection nearest cross street
- Route
- Mile post start
- Mile post end
- Node
- User ID
- X minimum
- X maximum
- Y minimum
- Y maximum

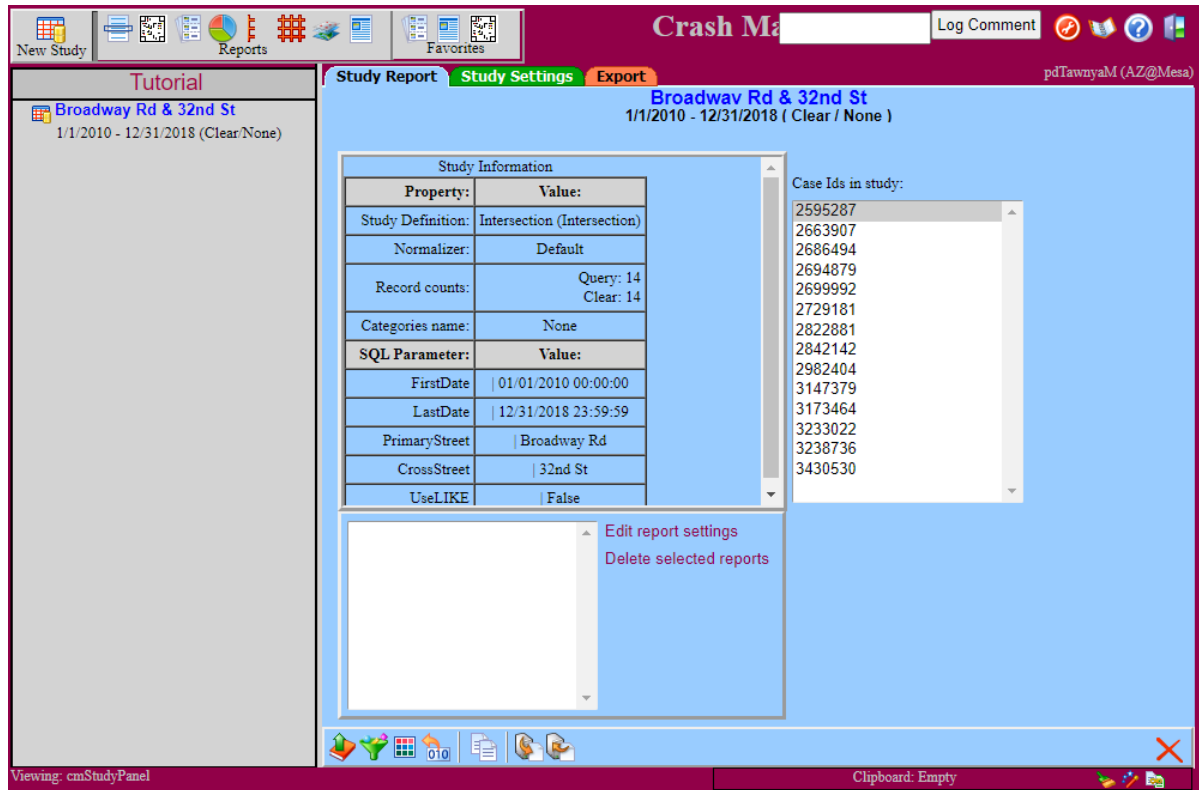
The fields displayed will change with the type of study selected.

The bottom of the panel can contains buttons for the study.

-  - [Category list](#)^[62] editor. This will allow the user to color collisions in the current study.
-  - Open the [filter](#)^[76] editor. This will allow the user to restrict the records shown in the study.
-  - Copy the current study(Reports under the current study will also be copied).
-  - Paste the current item in the clip board.
-  - Load a study [template](#)^[143].
-  - Save the current settings as a study [template](#)^[143].
-  - Re-query study. This causes Crash Magic to reissue the study query used to collect the collision records. This button will rarely be used, as the program knows when to refresh its content based on user actions. However, in some cases, you may need to give it that extra nudge. For example, if the contents of the crash database change due to the actions of another program (i.e. data entry) it will be necessary to tell Crash Magic to refresh its query.
-  - Export collision records.
- The current project. This drop down menu allows the user to select other projects to move the study to. Unlike the copy and paste buttons the study is moved from the project.

-  - [Delete button](#) ⁶⁶

Study info tab



The Study Info tab provides information on the data retrieved by a study

- **Study Definition** - Name of the study type used. This query can be found in the .shared user under the query section.
- **Filter name** - The name of any filter that is currently applied to the study.
- **SQL record count** - The number of collision records returned by the study.
- **Filtered record count** - The number of collision records after a filter has been applied.
- **Case Ids in study** - This is a list of the case ids that have been found the study.
- **Reports in this study** - This window shows the reports under the current study.
- **Delete selected reports** - This link allows any report selected in the "Reports in this study:" window to be deleted.
- **Edit reports settings** - This link will open the report settings tab of the first item selected in the "Reports in this study:" window.

The bottom buttons are will be the same as the [study settings tab buttons](#) ¹³⁸.

Study types

A study type specifies the means by which data can be gathered for analysis. Each study type requires a specific list of parameters that the study will use to get collision records from the database. This section lists and describes each of the available study types and the .

While a study has a few parameters that are used for data selection, that's just the first step. After the data has been gathered from the database, the Crash Magic [filter](#)^[76] can be utilized to further limit the selected records. So, you can select an intersection study and gather all the crash records that have matching primary and cross streets. But from there, you can apply a simple filter to indicate that you are only interested in crashes that were within 100 feet of the intersection. You can even have the filter check to see if the crash was a rear-end collision, in which case 200 feet is acceptable. The filters are extremely powerful and can be as complex you want to make them.

Note that many of the study types lend themselves to multi-purposing through the use of study templates. So, for example, you may refer to each of your intersections with a "Facility Id", and you may be surprised to find that Crash Magic specifies "node" as the study type in your study template(s). This flexibility enables Crash Magic to handle almost any sort of data and fields you have available.

Address

The address study type is used for retrieving data on a specific street between one address and another. This study type is very useful if your crash data contains an address field that specifies an appropriate value for mid-block collision locations. It is somewhat less useful if your data only includes addresses that map to the nearest intersection.

The address study type fully supports [street aliases](#)^[50] when available.

The parameters you'll need to provide for this study type are:

- Start date
- End date
- On street name
- Begin address
- End address

All Data

The all data study type is special in that it does not gather data for a specific geographic location. The only parameter this study type accepts is a date range. This makes the study useful for aggregated system-wide reporting such as high crash location lists, injury severity analysis, monthly trend analysis, etc. In addition, the ad-hoc nature of this study type makes it useful for querying data for which a study type is not otherwise available. Using the [filter mechanism](#)^[76], the data returned from this query may be

limited to a particular crash type, roads with a certain attribute, drivers with specific attributes, etc. Creative use of filters within an all data study can even identify streets, intersections, zip codes or other geographic regions.

The parameters you'll need to provide for this study type are:

- Start date
- End date

Case ID

The Case Id study type is used when a specific list of crash records need to be analyzed or reported on. A Crash Magic system table is used to store a list of case id numbers and other identifying information. Those crashes are tied to a specific study in the project tree. This type of study is used almost exclusively by 3rd party or in-house systems that gather data for analysis in Crash Magic. Those systems can make a MAGICAUTO or SOAP call that populates the list of case id's and creates the desired study and report(s).

One specific use of the Case Id study type is the Map Magic plug-in for ArcGIS. When a user selects a group of crashes in ArcGIS and uses Map Magic to report on those crashes, it utilizes the Case Id study type.

This study type does not accept any user input on the study form.

Intersection

The intersection study type is probably the most commonly used. The user provides a date range as well as a primary and cross street. All crashes referenced to that location are included in the study. When configured properly, primary and cross street names are interchangeable.

The intersection study type fully supports [intersection aliases](#)^[59] when available.

The parameters you'll need to provide for this study type are:

- Start date
- End date
- Primary street name
- Cross street name

Node

The node study type has two common uses. The first is for state or county road departments that have organized their systems into nodes and links. The second is for cities who have assigned a unique identifier to specific locations (like intersections) on their roadway system. This method of locating crashes can greatly improve the accuracy and efficiency of a study. This is because there are no chances of spelling

mistakes or duplicate location names. Nodes can also be useful for identifying related individual parts of a larger traffic control location. (such as two divided boulevards intersecting)

The node study type fully supports [node aliases](#)⁶¹ when available.

The parameters you'll need to provide for this study type are:

- Start date
- End date
- Node ID

Route Milepost

The Route milepost study type is used for retrieving data on a specific street between one milepost and another. This study type is very useful if your crash data contains an address field that specifies an appropriate value for mid-block collision locations. It is somewhat less useful if your data only includes addresses that map to the nearest intersection.

The route milepost study type fully supports [street aliases](#)⁵⁰ when available.

The parameters you'll need to provide for this study type are:

- Start date
- End date
- On street name
- Start milepost
- End milepost

User CID

User CID studies are special studies where collision have already defined for a specific location in the data. The CID study allows users to pass the parameters to identify these locations. This study can accept one or two parameters to identify the location.

The parameters you will need to provide for this study type is:

UserCID1

If required:

UserCID2

XY Rectangle

The xy rectangle study type is used for retrieving data within a rectangular area defined by a pair of coordinates. (two corners of the rectangle) This study type is typically used with data collected by GPS. The xy rectangle study is also used by the Map Magic for ArcGIS plugin.

The parameters you'll need to provide for this study type are:

- Start date

- End date
- Minimum X, Minimum Y coordinate
- Maximum X, Maximum Y coordinate

Corridor

The corridor study type is essentially a sequence of intersection studies tied together with additional parameters to indicate the scope of the query. The corridor study requires that a street layout table be populated first.

The corridor study type fully supports [street aliases](#)^[50] and [intersection aliases](#)^[59] when available.

The parameters you'll need to provide for this study type are:

- Start date
- End date
- Primary street name
- First cross street name
- Last cross street name
- Maximum distance from intersection along cross streets
- Minimum distance from intersection along primary street
- Maximum distance from end points of corridor along primary street

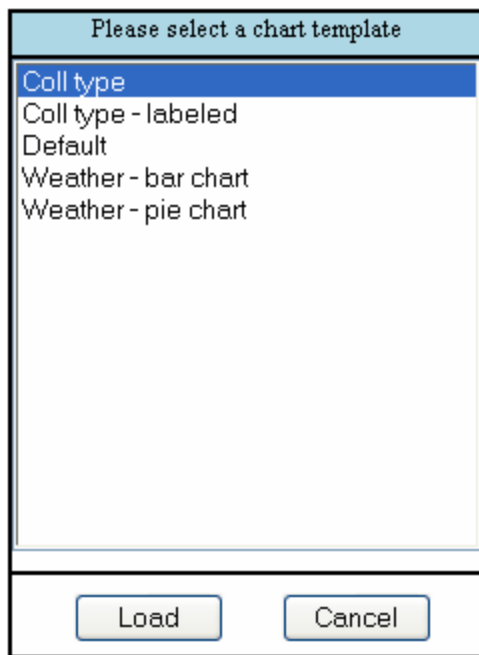
4.13 Templates

When creating projects, studies and reports, it will often be desirable to save the settings you've created. Templates make this possible, and even easy!

Load template



The **load template button** raises this dialog to apply a template to the current project, study or report. Think of this as "getting a template from the server".



Note that one of the template is called "Default". This is the template that will be loaded automatically when the current type of project, study or report is loaded. For example, create a chart for "weather", make it a pie chart with slices in shades of blue, with labels and percentages. Save this as "Default" and the next time you generate a new chart, it will be just like this template.



The **save template button** raises this dialog to save a template based on the current project, study or report. Think of this as "putting a template on to the server".

Please choose a name for this chart template

Existing templates:	Enter a new template name:
<ul style="list-style-type: none">Coll typeColl type - labeledDefaultWeather - bar chartWeather - pie chart	<input type="text" value="Coll type"/>
	<input type="button" value="Create a new template"/>
<input type="button" value="Replace existing template"/>	
<input type="button" value="Delete selected template"/>	
<input type="button" value="Cancel"/>	

Save template

Enter topic text here.



5 Administration

5.1 Overview

Crash Magic has been divided into three sections: Analysis, Data Entry, and Administration. The intent here is to keep the administrative functions out of the way of average users that don't need to know about the administration functionality.

The first part of this chapter, Getting Started, describes installing and configuring the program. The remainder of the chapter is a description of what is provided on the "Administration side" of the program.

5.2 Data

The data used to generate Crash Magic diagrams and reports is any data collected for individual crashes and stored in an SQL database. A large number of jurisdictions presently enter the necessary data into a centralized database for use by engineers and enforcement officials. CM requires only that data be stored in a compatible SQL database. Some common SQL compatible databases are Microsoft SQL Server™; Oracle™; and PostgreSQL™. Crash Magic supports fully normalized relational data; "flat" databases; and everything in-between. Crash Magic does not require a specific table structure or data dictionary. Crash data is read "in-place" in read only mode.

Pd' Programming usually performs the configuration steps necessary for your data to be read by Crash Magic. The Configuration chapter provides a detailed description of how to prepare the required settings, queries and other information required for Crash Magic to read your data.

5.3 Installation

Crash Magic Online is a server based application. This means that the actual software resides on a network server, usually a different computer than the user sits in front of.

Crash Magic Online can be installed as a Windows service application, or as an ISAPI plugin to Microsoft IIS. In either case, the first step is to install the program. Once the program is installed, a maintenance form is presented that guides the user through the steps to prepare the application for use. After this step, the "group administrator" can access the program to specify custom configurations and default settings.

The installation of Crash Magic is a four step process. In large organizations, these steps will typically be distributed across different roles or even departments. In smaller organizations, the steps may be performed by as few as a single person.

- **Project Manager** - The person in charge of having the software made available for use is the project manager. This person was likely involved in the procurement

process, and communicates with all the stakeholders and support personnel to implement the product for the agency. This person should be in close contact with Pd' Programming for support.

- **Database administrator** - This person has full access to logins, roles, permissions, databases, tables, etc. as they relate to the Crash Magic system. At a minimum, this includes the Crash Magic system tables as well as the agency's crash database to be analyzed by Crash Magic.
- **Server administrator** - This person is in charge of the physical or virtual server that will host the Crash Magic software. The role of installing the software and any needed drivers, verifying system requirements and managing IIS if used falls to this person. This is the person who will have/need license keys, master admin password, etc.
- **Group administrator** - This is a Crash Magic role. Once the database is configured, and the software installed, the Crash Magic group administrator takes over and performs the final configuration to make the software work as desired with necessary defaults, permissions, etc. The group administrator will also be the ongoing first point of contact for users of the system, and will be the person most likely to contact Pd' Programming with support questions. Suggested pre-requisites for this role are:
 - Knowledge of the crash database elements
 - Knowledge of the analysis methods used by the organization
 - Some experience or training with Crash Magic

The current hardware and software requirements for Crash Magic can be found at:
<https://www.pdmagic.com/kb/CurrentCrashMagicRequirements>

1. Project manager steps

Pre-installation Tasks

The following tasks should be done before installing Crash Magic:

- **Review the hardware software requirements**
<https://www.pdmagic.com/kb/CurrentCrashMagicRequirements>
- **Crash Magic server** software and updates
 1. Install any Windows updates to the server that will be running Crash Magic. This includes and .NET updates.
 2. Install database drivers on the server that will be running the Crash Magic software(You should test establishing a connection to the database server using the database tools that install with the driver. Refer to documentation on your specific database driver.)
- **Crash Magic server.** Gather information required for the server that will run Crash Magic
 1. Machine name of the server. (Also public name, if different)
 2. IP address of the server. (Also public IP, if different)

3. Server administrator log-in and password to configure IIS and to install the Crash Magic program
- **SYS DB server.** Gather information required for the database or schema that will store the Crash Magic [system](#) tables¹⁸⁵
 1. Name of the database server that will store the Crash Magic system tables, as the Crash Magic server will see it.
 2. IP address of the database server
 3. Administrator log-in and password to the database that can create the Crash Magic system tables. (DDL access)
 - **Crash DB server.** Gather information required for your crash database or schema
 1. Name of the server that your crash data is stored on, as the Crash Magic server will see it.
 2. IP address of the server
 3. Database login and password to access your crash data
 4. Database password
 5. Oracle environments that restrict user sessions will also need to ensure that users have been granted enough sessions for the program to function correctly. While Crash Magic does reuse Oracle sessions, some user experiences require multiple Oracle sessions. Pd' Programming recommends a minimum of five oracle sessions per Crash Magic user.
 - Download the Crash Magic installation software from www.GovernmentTools.com
 - Download the XML configuration for Crash Magic for your agency. This configuration file will be imported into the system tables, and tells Crash Magic how to read the collision data. If creating a second instance of an existing server, this information may be exported from the existing system.
 - Request license information from Pd' Programming (help@pdmagic.com)
 1. Jurisdiction name
 2. License key for Crash Magic
 3. Pd' Programming assigned client ID
 4. User Group name
 - Confirm that you have sufficient hard drive space to install Crash Magic.

By default, Crash Magic will not allow a second instance of itself (.exe or .dll) to be run on a single server. To override this, add the following line in the cmVirtualRegistry.ini file after installation:

AllowMultipleInstances=true

The cmVirtualRegistry.ini file can be found in the Sys folder under the Crash Magic Program Data folders.

2. Server administrator steps

Install IIS roles and features

Review the hardware software requirements:

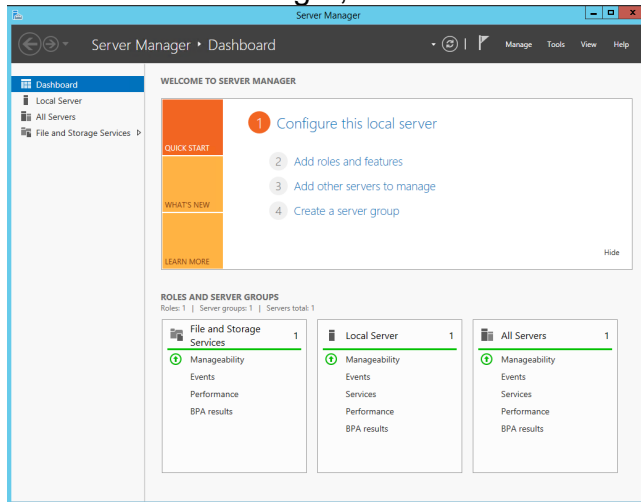
<https://www.pdmagic.com/kb/CurrentCrashMagicRequirements>

Note: When Crash Magic Online is installed as an ISAPI plug-in, the port and IP address are no longer relevant. This information is managed by the web server. In addition, multiple instances may be installed, and the application may be accessed from any virtual directory on the server. (i.e. <http://myserver/myvirtualdir/cmo.dll>)

Important, please start the install process by completing all the [Pre-installation tasks](#) ¹⁴⁸.

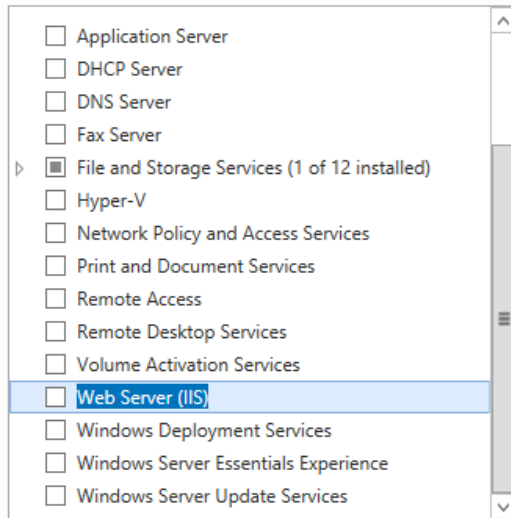
In order to run an ISAPI application in IIS, the following steps must be taken.

1. Launch Server Manager, and select "Add roles and features"

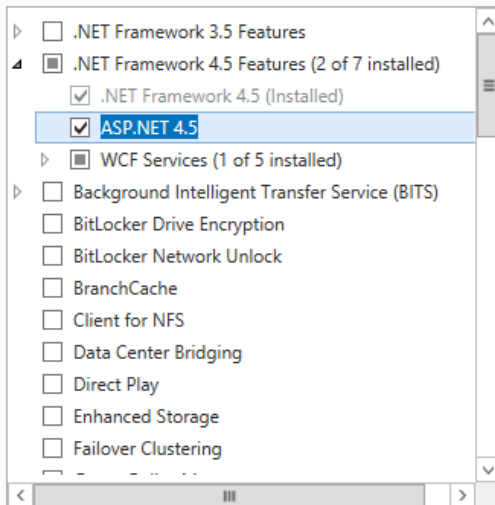


2. Click "Next" on the "Before you begin" page.
3. Select "Role-based or feature-based installation" on the "Select installation type" page, and click "Next".
4. On the "Select destination server" page, choose "Select a server from the server pool" and be sure the proper server is selected in the list. Click "Next".

5. On the "Select server roles" page, check the "Web Server (IIS)" role. Select "Add Features" to have the appropriate features added. Select "Next".



6. On the "Select features" page, select .NET Framework 4.5 Features / ASP.NET 4.5 (or latest) and press Next.



7. On the "Web Server Role (IIS)" page, select "Next".
8. On the "Select role services" page, in addition to the defaults, select the following options:

<ul style="list-style-type: none"> <input checked="" type="checkbox"/> Web Server <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Common HTTP Features <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Default Document <input checked="" type="checkbox"/> Directory Browsing <input checked="" type="checkbox"/> HTTP Errors <input checked="" type="checkbox"/> Static Content <input checked="" type="checkbox"/> HTTP Redirection <input type="checkbox"/> WebDAV Publishing <input checked="" type="checkbox"/> Health and Diagnostics <ul style="list-style-type: none"> <input checked="" type="checkbox"/> HTTP Logging <input type="checkbox"/> Custom Logging <input type="checkbox"/> Logging Tools <input type="checkbox"/> ODBC Logging <input type="checkbox"/> Request Monitor <input type="checkbox"/> Tracing <input checked="" type="checkbox"/> Performance <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Static Content Compression <input checked="" type="checkbox"/> Dynamic Content Compression <input checked="" type="checkbox"/> Security <ul style="list-style-type: none"> <input checked="" type="checkbox"/> Request Filtering <input type="checkbox"/> Basic Authentication <input type="checkbox"/> Centralized SSL Certificate Support <input type="checkbox"/> Client Certificate Mapping Authentication <input type="checkbox"/> Digest Authentication <input type="checkbox"/> IIS Client Certificate Mapping Authentication <input type="checkbox"/> IP and Domain Restrictions <input type="checkbox"/> URL Authorization <input checked="" type="checkbox"/> Windows Authentication 	<ul style="list-style-type: none"> <input checked="" type="checkbox"/> Application Development <ul style="list-style-type: none"> <input type="checkbox"/> .NET Extensibility 3.5 <input checked="" type="checkbox"/> .NET Extensibility 4.5 <input type="checkbox"/> Application Initialization <input type="checkbox"/> ASP <input type="checkbox"/> ASP.NET 3.5 <input checked="" type="checkbox"/> ASP.NET 4.5 <input type="checkbox"/> CGI <input checked="" type="checkbox"/> ISAPI Extensions <input checked="" type="checkbox"/> ISAPI Filters <input type="checkbox"/> Server Side Includes <input type="checkbox"/> WebSocket Protocol <input type="checkbox"/> FTP Server <ul style="list-style-type: none"> <input type="checkbox"/> FTP Service <input type="checkbox"/> FTP Extensibility <input checked="" type="checkbox"/> Management Tools <ul style="list-style-type: none"> <input checked="" type="checkbox"/> IIS Management Console <input type="checkbox"/> IIS 6 Management Compatibility <input checked="" type="checkbox"/> IIS Management Scripts and Tools <input type="checkbox"/> Management Service
--	--

- a. "Common HTTP Features / HTTP Redirection"
 - b. "Performance / Dynamic Content Compression"
 - c. "Security / Windows Authentication" (optional - used for AD integration)
 - d. "Application Development / ASP.NET 4.5" (and its required features)
 - e. "Application Development / ISAPI Extensions"
 - f. "Application Development / ISAPI Filters"
 - g. "Management Tools / IIS Management Scripts and Tools"
9. Click "Install" on the "Confirm installation selections" page.
 10. Reboot when the process is complete.

Crash Magic Online Install and Configuration Tool

Be sure to review the current Crash Magic requirements:

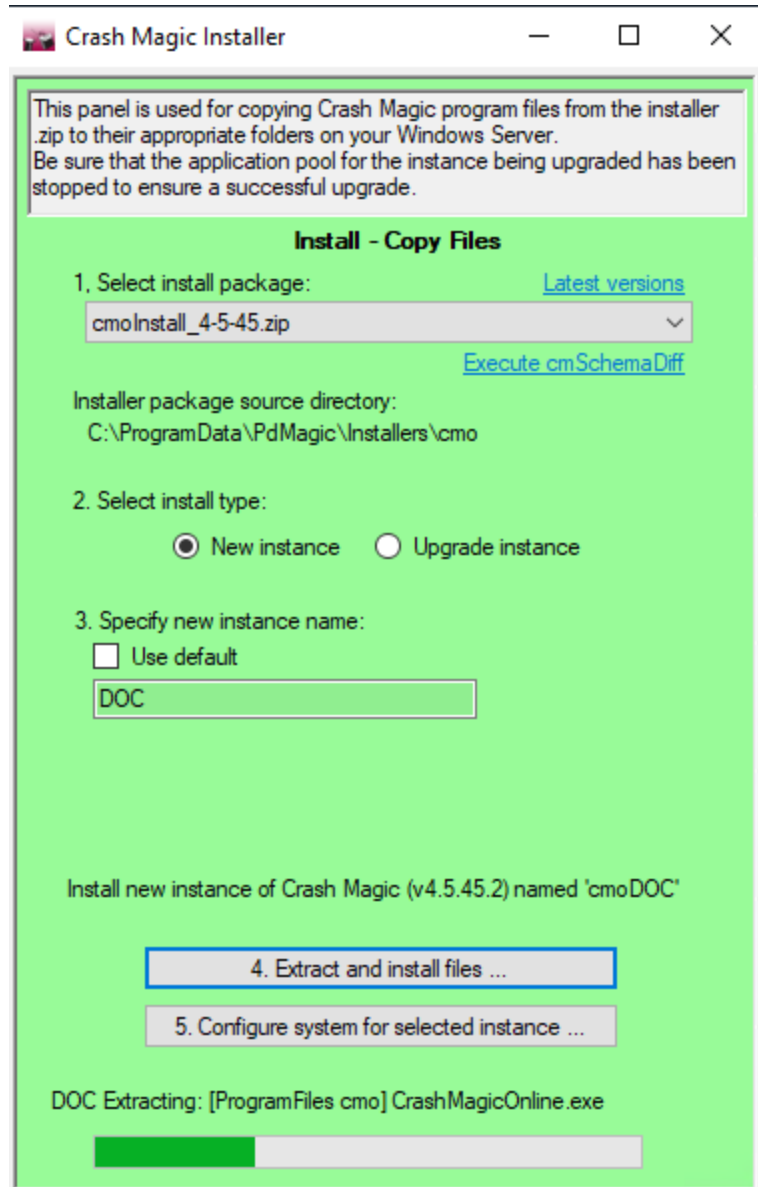
<https://www.pdmagic.com/kb/CurrentCrashMagicRequirements>

Installation of Crash Magic begins by downloading the Crash Magic Installer program from www.GovernmentTools.com. (locate the "Installers" link) Run this setup program on the server. A new shortcut entitled "Crash Magic Installer" will be added to the system menu and to the desktop.

Note: The current installer handles all of the IIS configuration details. However, if you prefer to take the steps yourself or would like to see what configuration steps will be executed, view the [manual steps](#)^[158] and the [sample install log](#)^[166].

Install the Crash Magic files

Start the process by launching the Crash Magic Installer from the start menu. ***Important, this utility must be run "as administrator".***



1. The first step will be to download the desired version of the Crash Magic program. This is done with the "Latest versions" link. Choose the desired package and download it. That package will be automatically selected in the list.
2. In step 2, choose "New instance" unless you are upgrading an existing instance of the program.
3. Next, choose an instance name. While "use default" is an option, we've found it to be cleaner to always specify an instance name. Some good options are "prod" or "dev" or "test". "cm" will also work. For this documentation, we are using the name "DOC".

This will be the prefix or suffix for the the names of the app pool, user accounts, folders and the end of the URL path used to access the system.

4. Next, select "Extract and install files...". You will be prompted to confirm before the copy starts. The process will validate the install package and then copy files from it into your Windows System; Program Files (x86)/PdMagic; and Program Data folders. This can take several minutes.
5. Once the file copy is complete, select the "Configure system for selected instance ..." button to open the configuration dialog.

Configuring Windows and IIS for this instance

Pre-load settings tab

System configuration settings
DOC
v4.5.45.2 @ C:\Program Files (x86)\PdMagic\cmoDOC

Pre-load settings | IIS Configuration | GIS IIS Configuration

This page is used for configuring the IIS Server to host Crash Magic as an existing or new site and/or application.

C:\ProgramData\PdMagic\cmoDOC\cmVirtualRegistry.ini

General properties

Install login: Install password:

Instance URL:

Service properties (does not apply to IIS installations)

Server port: Server IP:

Special properties

Allow multiple instances: ☐ *Only select if this server will hosted multiple instances*

Ignore master update: ☐ *ONLY for debug purposes*

Status [Clear](#)

Ready

Done

1. Create a login and password for the "Install login". This will be needed to access the program "maintenance" when the application is not yet fully configured. Enter the "Instance URL". This value is provided to the browser to access some aspects of the program. It is important that the URL be the value that a user will see when running the program. It may or may not be the machine name. This value can be updated after the program installation is complete if the URL is not yet known

2. For an IIS installation, skip the server port and server IP options, as IIS will handle these.
3. If this server will host multiple instances of Crash Magic, then all instances must have the "Allow multiple instances" checkbox checked. Do not check this box unless you are certain there will be multiple instances. It removes some protective error handling from the launch process.

IIS Configuration tab

Crash Magic Online Install and Configuration Tool v1.4.3.0

System configuration settings DOC

v4.5.45.2 @ C:\Program Files (x86)\PdMagic\cmoDOC

Pre-load settings IIS Configuration GIS IIS Configuration

This page is used for configuring the IIS Server to host Crash Magic as an existing or new site and/or application. IMPORTANT - form will be blank even if instance is already configured. Be sure that IIS is installed with HTTP Redirect and both ISAPI and .NET features included.

New Instance Configuration Steps

Account and Folder Security

Group:

User:

Password:

Password:

IIS Site Settings

4. Select web site:

5. Select application:

6. Configure site and app:

7. Select application pool:

8. Configure app pool:

Status [Clear](#)

```
ADD APPLICATION
+ Site / Path: (Default Web SiteC:\Program Files (x86)\PdMagic\cmoDOC\Bin\ISAPI)
+ Application: (/cmDOC)
ADD APPLICATION POOL
+ Pool Name: (cmISAPI32DOC)
```

Ready

1. It is recommended that you accept the default Group and User account names. These accounts will be created as local Windows accounts. They will be used by the IIS application to access the local file system. (ProgramData/PdMagic/cmoDOC and Program Files/PdMagic/cmoDOC/bin/ISAPI) Select an appropriate password.
2. Click "Create this user and group". Any errors will appear in red in the Status log at the bottom.
3. Click "Give this user required access". This is the step that will iterate all the files that the program will need to access. It will take a minute or so to complete.
The optional "Create local cmREST user" creates a local account called cmREST. This account can be used when the system is configured for Active Directory logins.

Rather than having to use an existing AD account, the local cmREST account credentials can be used to get past IIS AD security. To enable this account, enter a password and click "optional: Create local cmREST user". This process can also be completed manually at a future date if needed.

4. In most cases, especially on dedicated servers / VMs, it is best to use the "Default Web Site" created by IIS. Other web sites can be created if the server is multi-homed or is configured for such.
5. A new application should always be created when installing a new instance. Click the "New ..." button and accept the default name.
6. Click the "Click to configure site" button and verify that there were no errors in the status list. This option sets all of the necessary options for the selected website and application.
7. A new application pool should always be created for a new Crash Magic instance. Click the "New ..." button and select the default application name. That application pool will be configured to use the cmUser account from step 1.
8. Click the "Click to configure pool" button and verify that no errors appear in the status box.

GIS IIS Configuration tab

In most cases, Crash Magic will be configured to interact with an ESRI or other GIS server. Even if not, it is wise to set up the required Crash Magic GIS services in IIS. Choose the GIS IIS Configuration tab.

System configuration settings
DOC
v4.5.45.2 @ C:\Program Files (x86)\PdMagic\cmoDOC

Pre-load settings | IIS Configuration | GIS IIS Configuration

This page is used for configuring the IIS Server to host PdGIServices as an existing or new site and/or application.

New Instance Configuration Steps

<p>Account and Folder Security</p> <p>Group: <input type="text" value="cmUserGroupDOC"/></p> <p>User: <input type="text" value="cmUserDOC"/></p> <p>Password: <input type="password" value="*****"/></p> <p><input type="button" value="2. Create this user and group"/></p> <p><input type="button" value="3. Give this user required access"/></p>	<p>IIS Site Settings</p> <p>4. Select web site: <input type="text" value="Default Web Site"/></p> <p>5. Select Crash Magic app: <input type="text" value="Default Web Site/cmDOC"/></p> <p>GIS app installed to: /cmDOC/services/pdgisservices</p> <p>6. Configure GIS app: <input type="button" value="Click to configure app"/></p> <p>7. Select application pool: <input type="text" value="cmISAPI32DOC"/> <input type="button" value="New ..."/></p> <p>8. Configure app pool: <input type="button" value="Click to configure pool"/></p>
---	---

Status [Clear](#)

Ready

1. Leave the Group and User the same as the IIS Configuration tab. Enter the same password from the IIS Configuration tab as well.
2. Skip the step of creating the user and group as it was just created in the prior section.
3. Click the "Give this user required access" button.
4. Select the same website as used in the IIS Configuration tab. *Note: if either this or the next option is not available, select "Done" and then "Configure system for selected instance..." again to re-open the dialog. They should now be available.*
5. Select the same Crash Magic app as used in the IIS Configuration tab.
6. Select "Click to configure app"
7. Choose "New..." to create a new application pool for the Pd GIS Services.

At this point, if you selected all the defaults, the program should be ready to run.

Other considerations

- If you made selections that resulted in one or more new web sites, you'll need to use the IIS Manager to adjust the bindings so that traffic is routed properly.
- If you are setting the system up for https, use the IIS Manager to install the appropriate certificate and bindings for port 443.

- If you are setting the system up for Active Directory authentication, that will require setting the "Authentication" option for the application to disable Anonymous Authentication and enable Windows Authentication. We've also found that it helps to verify that the only Provider for the Windows Authentication is "NTLM". Remove any others for this application. (This seems to important for the MS Edge browser)
- If any errors occurred during the installer run (they'll be shown in red in the status window) be sure to figure them out before attempting to run the program.

The steps that follow are now all handled by the Crash Magic Installer utility. This documentation is here to provide information to system administrators about the steps that are taken in that process. Pd' Programming does not recommend using these steps to install and configure Crash Magic on IIS

To configure Crash Magic Online to run inside of IIS 7 as an ISAPI plugin, the following steps must be taken. These steps assume that your server and network are correctly configured.

INSTALL IIS and components If not already installed

1. Install IIS while logged in as administrator to the server
2. Click Start/All Programs/Administrative Tools/Server Manager
3. Click the Roles
4. With the Roles summary page open click Add Roles
5. On the Select Server Roles window check the box next to Web Server (IIS)
6. Click the Next button
7. On the Role Services window check the box next to "CGI" under Web Server/Application Development
8. Check the box next to "ISAPI Extensions" under Web Server/Application Development
9. Check the box next to "Dynamic Content Compression" under Web Server/Performance
10. Check the box next to IIS "Management Scripts and Tools" under Management Tools
11. Click the Next button
12. Click the closed button when complete

Additional Windows updates should be installed at this time if required

Create cmUser to run the Crash Magic application in IIS

1. Open the the server manager (You may wish to move the Post install utility window to the bottom of the screen)
2. Open Configuration/Local Users and Groups/Users
3. Right click in the Users window and select "New User..."

4. Enter a user name of cmUser for the user name
5. Full name Crash Magic user
6. Description of Crash Magic IIS user
7. Enter a password and confirm the password(This password will be needed for future steps)
8. Uncheck the "User must change password at next logon"
9. Check User cannot change password
10. Check Password never expires
11. Click the create button
12. Click the close button

Create the cmUsers group for Crash Magic

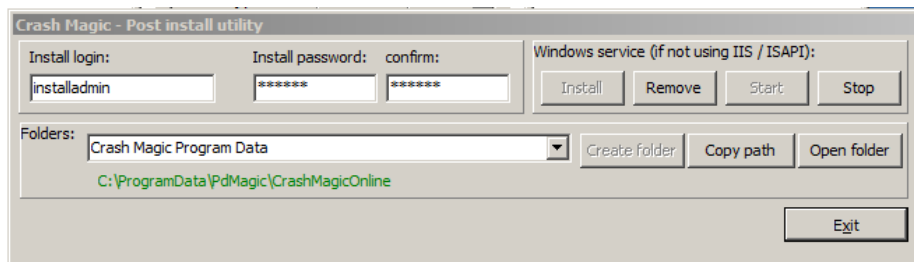
1. Create a user group for Crash Magic:
2. Ensure the Computer Management window is still open from the previous step
3. Right click on the Groups folder under Local User and Groups
4. Select New Group...
5. Enter a Group name of cmUsers
6. Enter a description of Crash Magic Online users
7. Click the Add button
8. Click the Locations... button
9. Select the local machine
10. Click OK
11. Enter cmUser in the object names to select field
12. Click the OK button to return
13. Click the Create button to create the group
14. Click the Close button

Make cmUser a member of the IIS_IUSRS group

1. Right click on the cmUser that was just created
 2. Select properties
 3. Click on the Member of tab
 4. Click the Add button
 5. Select the local computer name from the Locations
 6. Enter IIS_IUSRS in the object names to select box
 7. Click the check names button to add the machine qualification(Machine Name>\cmUser)
 8. Click the OK button
- * DB2 database users will need to make cmUser a member of the DB2USERS group for Crash Magic to access the DB2 database drivers.

Grant permissions to the Crash Magic Sys data folders

1. Return to the Crash Magic post-install utility(If it has been closed select Start/All Programs/Pd Magic/Crash Magic post-install utility. You may wish to leave this utility open for the remainder of the install process.)
2. In the folder section select the Crash Magic Program Data



3. Click the Open folder button
4. With the Crash Magic program data folder open right click on the Sys folder shown and select Properties
5. Click on the Security tab
6. Click the Edit button
7. Click the Add button
8. Select the local machine for the location
9. Enter cmUsers in the object names to select box
10. Click the check names button to add the machine qualification
11. Click the OK button to return to the Permissions for window
12. With the cmUsers selected ensure that List folder contents, Read and Write are checked
13. Click the OK button and close the Permissions for window
14. Click the OK button to close the Properties window

Grant cmUsers permissions to the Crash Magic ISAPI folder

1. With the Crash Magic post-install utility application open
2. In the folder section select the Crash Magic Bin folder
3. Click the Open folder button
4. With the Crash Magic Bin folder open right click on the ISAPI folder and select Properties
5. Click on the Security tab
6. Click the Edit button
7. Click the Add button
8. Select the local machine for the location
9. Enter cmUsers in the object names to select box
10. Click the check names button to add the machine qualification
11. Click the OK button to return to the Permissions for window
12. With the cmUsers selected ensure that Read & execute, List folder contents and Read are checked
13. Click the OK button and close the Permissions for window
14. Click the OK button to close the Properties window

Create a new cmISAPI32 Application pool

1. Open the IIS manager
2. Expand the machine in the connections tree, and click on the Application pools in IIS
3. Click the Add Application Pool...
4. Enter a name of cmISAPI32
5. Use the default setting for .NET Framework version
6. Use the default setting for Managed pipeline mode of Integrated

7. Ensure the check box of Start application pool immediately is checked
8. Click OK to create the pool
9. Right click on the cmISAPI32 application pool for Crash Magic and select advanced settings.
10. 64 bit operating systems will need to set Enable 32-Bit Applications to True in the (General section)
11. Set the identity field to cmUser created from the previous steps
12. Set Idle time out to 360 min
13. Click OK to close the Advanced Settings window

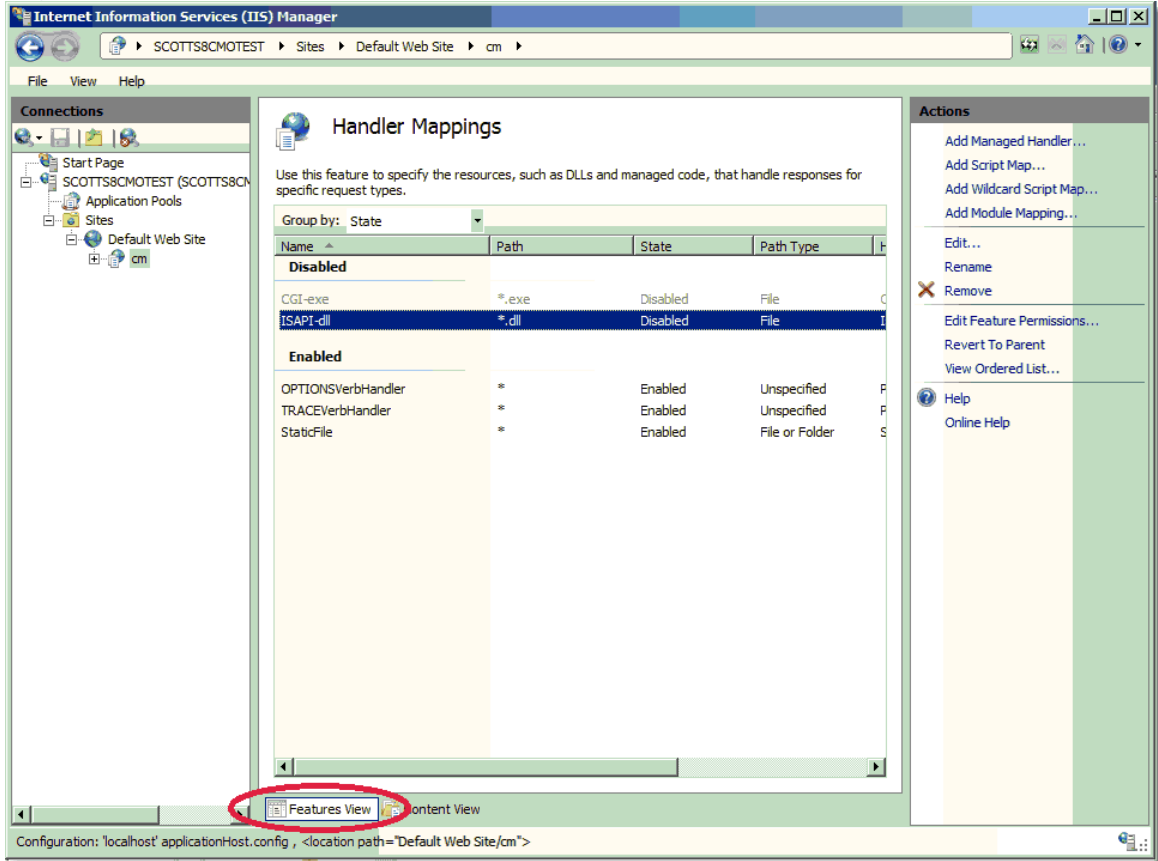
In IIS add the Crash Magic application to the Default web site

1. Expand the Sites section in the connections tree.
2. Right click on the Default Web Site and select Add Application
3. Enter an Alias for the application(cm is the recommended alias)
4. Select the application pool for Crash Magic(cmISAPI32)
5. With the Add Application window still open return to the Crash Magic post-install utility application
6. Select Crash Magic ISAPI in the Folders drop down menu
7. Click the Copy path button
8. Return to the Add Application window
9. Paste the path into the Physical path box
10. Click the Connect as button
11. Click the Specific user radio button
12. Click the Set button
13. Enter the cmUser in the user name field
14. Enter the password for cmUser in the Password field.
15. Enter the password for cmUser in the Confirm Password field
16. Click the OK button
17. Click the OK button
18. Click the Test Settings button to confirm cmUser can access the ISAPI directory
19. Click OK to close the Add Application window

Edit the ISAPI dll handler mappings

1. While on the Home screen for the application(/cm Home) that was just created, Ensure that Features View is selected.
2. Open the Handler Mappings icon

3. Right click on the ISAPI-dll and select Edit



4. With the Edit Module Mapping still open return to the Crash Magic post-install utility application
5. Select Crash Magic ISAPI dll in the Folders drop down menu
6. Click the Copy path button
7. Return to the Edit Module Mapping window
8. Paste the path into the Executable box
9. Click the request Restrictions button
10. Ensure that Invoke handler only if request is mapped to File checkbox is checked under the Mapping tab.
11. Under the verb tab ensure that All verbs radio button is selected
12. Under the Access tab ensure the Execute radio button is selected
13. Click OK to return the edit script map
14. Click OK to close the window
15. Click Yes to allow this ISAPI extension

Allow ISAPI executions

1. While still in Handler Mappings for the application Right click on ISAPI-dll and select Edit Feature Permissions...
2. Select edit feature permissions
3. Check the Execute check box
4. Click OK(This will also enable CGI-exe)

Add the CrashMagicOnline_ISAPI.dll as the default document

1. Click on the directory created for Crash Magic(cm) in the Connections section of the IIS manager to return to the IIS Home screen for the application(/cm Home)
2. Open the Default Document icon
3. Click Add under Actions
4. Enter CrashMagicOnline_ISAPI.dll in the Name field
5. Click the OK button to return

Add a virtual directory name CMFiles to the application

1. Right click on the directory created for Crash Magic(cm) under Connections section of IIS
2. Select add virtual directory
3. In the Alias text box enter CMFiles
4. With the Add Virtual Directory window still open return to the Crash Magic post-install utility application
5. Select Crash Magic SYS Files in the Folders drop down menu
6. Click the Copy path button
7. Return to the Add Virtual Directory window
8. Paste the path into the Physical path box
9. Click the Connect as button
10. Click the Specific user radio button
11. Click the Set button
12. Enter the cmUser in the user name field
13. Enter the password for cmUser in the Password field.
14. Enter the password for cmUser in the Confirm Password field
15. Click the Ok button
16. Click the OK button
17. Click the Test Settings button to confirm the CMUser can access the Files directory
18. Click the OK button to close the Add Virtual Directory window

Add another virtual directory named CMCache to the cm application

1. Right click on the directory created for Crash Magic(cm) under Connections section of IIS
2. Select add virtual directory
3. In the Alias text box add CMCache
4. With the Add Virtual Directory window still open return to the Crash Magic post-install utility application
5. Select Crash Magic Cache in the Folders drop down menu
6. Click the Copy path button
7. Return to the Add Virtual Directory window
8. Paste the path into the Physical path box
9. Click the Connect as button
10. Click the Specific user radio button
11. Click the Set button
12. Enter the cmUser in the user name field
13. Enter the password for cmUser in the Password field.
14. Enter the password for cmUser in the Confirm Password field

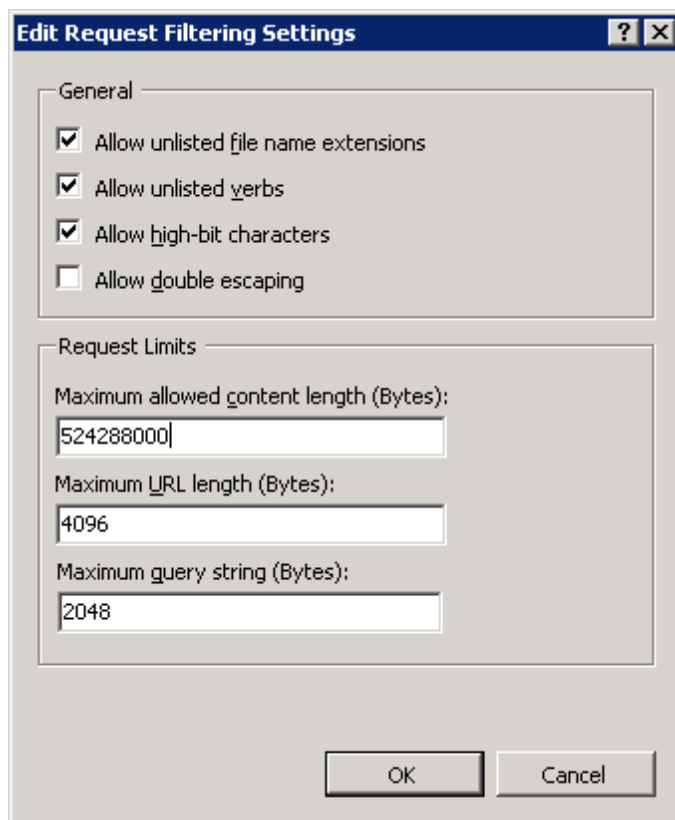
15. Click the Ok button
16. Click the OK button
17. Click the Test Settings button to confirm the CMUser can access the Files directory
18. Click the OK button to close the Add Virtual Directory window

Add MIME types(Depending on the web server some extensions may already be enabled)

1. Click on the directory created for Crash Magic(cm) in the Connections section of the IIS manager to return to the IIS Home screen for the application(/cm Home)
2. Double click MIME types in the Crash Magic site
3. Click the Add... link
4. Enter .svg for the File name extension box
5. Enter image/svg+xml for the MIME type box
6. Click OK to add the MIME type
7. Click the Add... link
8. Enter .xap for the File name extension box
9. Enter application/x-silverlight-app for the MIME type box
10. Click OK to add the MIME type
11. Click the Add... link
12. Enter .xaml for the File name extension box
13. Enter application/xaml+xml for the MIME type box
14. Click OK to add the MIME type

Increase the file import size for IIS

1. Click on the directory created for Crash Magic(cm) in the Connections section of the IIS manager to return to the IIS Home screen for the application(/cm Home)
2. Click on Request Filtering icon
3. Click Edit Feature Settings link
4. Change the Maximum allowed content length (Bytes) field from 30000000 to 524288000



5. Click the OK button to close the window

Stop and restart the application pool for Crash Magic

1. Click on the Application Pools in the Connections section of the IIS manager
2. Select the Crash Magic application pool(cmISAPI32) by clicking on it
3. Click the stop link under the Action section of IIS Manager
4. Click the Start link under the Action section of IIS Manager

Check the web site

1. Click on the directory created for Crash Magic(cm) under connections to return to the IIS Home screen for the application(/cm Home)
2. Click the Browse *:80(http) under the Action section of the IIS Manager to test the web site and see the Crash Magic Online Maintenance Form open

Ensure the Install Administrator login and password are set

1. Pass the URL for the site, the login and password information on to the person designated as the Database Administrator for Crash Magic
2. Pass the URL for the site, the login and password information on to the person designated as the Group Administrator for Crash Magic and instruct them to wait for notification from the database administrator before performing their installation tasks
3. Click the Exit button save the information and close the window

Upload Read Ahead

Ensure that the IIS Management Scripts and Tools role has been installed on the server.

Open a command prompt window as Administrator and execute the AppCmd.exe command with the appropriate parameters will set the packet size to a value that should work well for Crash Magic SOAP needs. The AppCmd.exe can be found in the %systemroot%\system32\inetsrv\ directory. While in the inetsrv directory run the following command replacing <Crash Magic Site Identification> with the identifier for the Crash Magic site (The default site id is "Default Web Site/cm").

Syntax: appcmd.exe set config "<Crash Magic Site Identification>" -section:system.webServer/serverRuntime /uploadReadAheadSize:"204800" /commit:apphost

```
appcmd.exe set config "Default Web Site/cm" -section:system.webServer/serverRuntime /uploadReadAheadSize:"204800" /commit:apphost
```

Other settings:

ApplicationPool RegularTimeInterval=0

Here is what the status log will show after following the normal IIS configuration steps.

```
CREATE USER AND GROUP
* Created user group (cmUserGroupDOC)
* Created user (cmUserDOC)
* Added user (cmUserDOC) to group (cmUserGroupDOC)
* Added user (cmUserDOC) to group (IIS_IUSRS)
ASSIGN FILE AND FOLDER SECURITY
* Gave group cmUserGroupDOC modify/list/read/write permissions to cmFolderRemap.txt
* Gave group cmUserGroupDOC modify/list/read/write permissions to cmURLRemap.txt
* Gave group cmUserGroupDOC modify/list/read/write permissions to cmVirtualRegistry.ini
* Gave group cmUserGroupDOC modify/list/read/write permissions to CrashMagic.xml
* Gave group cmUserGroupDOC modify/list/read/write permissions to C:\ProgramData\PdMagic\cmoDOC\Publications
* Gave group cmUserGroupDOC read & execute permissions to C:\Program Files (x86)\PdMagic\cmoDOC\Bin\ISAPI\cmo.d
CREATE CMREST USER
ADD APPLICATION
+ Site / Path: (Default Web SiteC:\Program Files (x86)\PdMagic\cmoDOC\Bin\ISAPI)
+ Application: (/cmDOC)
ADD ISAPI HANDLER
* Site / Instance / Path: (Default Web Site / DOC / /cmDOC)
* Set system.webServer/handlers/accessPolicy to (Read, Script, Execute)
= Handler: (Default Web SiteDOC_Handler) already exists
NEW ISAPI CGI RESTRICTION
* Section: (system.webServer/security/isapiCgiRestriction)
= An element already exists that points to (C:\Program Files (x86)\PdMagic\cmoDOC\Bin\ISAPI\cmo.d
SET APPLICATION USER
* Application: (/cmDOC)
* Site: (Default Web Site)
+ Set userName: (cmUserDOC)
+ Set password: (***)
SET MIME TYPES
* Section: (system.webServer/staticContent)
* Site / Path: (Default Web Site/cmDOC)
```

```

= Mime entry already exists for (.svg)
= Mime entry already exists for (.xap)
= Mime entry already exists for (.xaml)
= Mime entry already exists for (.pdxml)
SET DEFAULT DOCUMENT
* Section: (system.webServer/defaultDocument) Set enabled = true
* Site / Path: (Default Web Site/cmDOC)
= (cmo.dll) already exists as default
SET MAX ALLOWED CONTENT LENGTH
* Section: (system.webServer/security/requestFiltering) Set OverrideModeDefault to (Allow).
= requestLimits/maxAllowedContentLength already set to (524288000)
SET SERVER RUNTIME PARAMETERS
* Section (system.webServer/serverRuntime) Set OverrideModeDefault to (Allow).
= uploadReadAheadSize already set to (2147483647)
= uploadReadAheadSize already set to (4294967295)
ADD VIRTUAL FOLDERS
* Site / Path: (Default Web Site/cmDOC)
+ Added virtual directory (/CMFiles=C:\ProgramData\PdMagic\cmoDOC\Pub\files)
+ Added virtual directory (/CMCache=C:\ProgramData\PdMagic\cmoDOC\Pub\Default\Cache\$)
ADD EVENT LOG SOURCE
+ Log/Source already exist: (PdMagic/cmoDOC)
ADD APPLICATION POOL
+ Pool name: (cmISAPI32DOC)
ADD SITE TO APPLICATION POOL
+ Site: (Default Web Site) added to pool: (cmISAPI32DOC)
APPLICATION POOL SETUP
ADD SITE TO APPLICATION POOL
+ Site: (Default Web Site) added to pool: (cmISAPI32DOC)
PREPARE APPLICATION POOL
* Pool name: (cmISAPI32DOC)
+ Set ManagedRuntimeVersion: (v4.0)
+ Set ManagedPipelineMode: (Integrated)
+ Set Enable32BitAppOnWin64: (True)
+ Set AutoStart: (True)
+ Set ProcessModel.IdleTimeout: (06:00:00)
+ Set Recycling.DisallowoverlappingRotation: (True)
+ Set Recycling.PeriodicRestart.Time: (00:00:00)
+ Set Recycling.PeriodicRestart.Schedule: (03:00:00)
+ Set ProcessModel.UserName: (CMDEVSERVER\cmUserDOC) Password: (***)
+ Set IdentityType: (SpecificUser)

```

The GIS IIS Configuration will look like this:

```

GIS ASSIGN FOLDER SECURITY
* Gave group cmUserGroupDOC modify permissions to C:\Program Files (x86)\PdMagic\cmsvcDOC
ADD VIRTUAL FOLDERS
* Site / Path: (Default Web Site/cmDOC)
+ Added virtual directory (/services=C:\Program Files (x86)\PdMagic\cmsvcDOC)
+ Added virtual directory (/services/heatmapfiles=C:\ProgramData\PdMagic\cmsvcDOC\output\heatmapfi
ADD APPLICATION
+ Site / Path: (Default Web SiteC:\Program Files (x86)\PdMagic\cmsvcDOC\pdgiservices)
+ Application: (/cmDOC/services/pdgiservices)
ADD EVENT LOG SOURCE
+ Log: (PdMagic)
+ Source: (pdGisServicesDOC)
ADD APPLICATION POOL
+ Pool name: (pdGisServicesDOC)
ADD SITE TO APPLICATION POOL
+ Site: (Default Web Site) added to pool: (pdGisServicesDOC)
APPLICATION POOL SETUP
PREPARE APPLICATION POOL

```

```
* Pool name: (pdGisServicesDOC)
+ Set ManagedRuntimeVersion: (v4.0)
+ Set ManagedPipelineMode: (Integrated)
+ Set Enable32BitAppOnWin64: (True)
+ Set ProcessModelUserName: (ThisServer\cmUserDOC) Password: (***)
+ Set IdentityType: (SpecificUser)
```

3. Database administrator steps

Create system tables

Crash Magic requires a database connection to store internal system data. These instructions show how to create the database tables. Please ensure that all required database drivers have been installed.

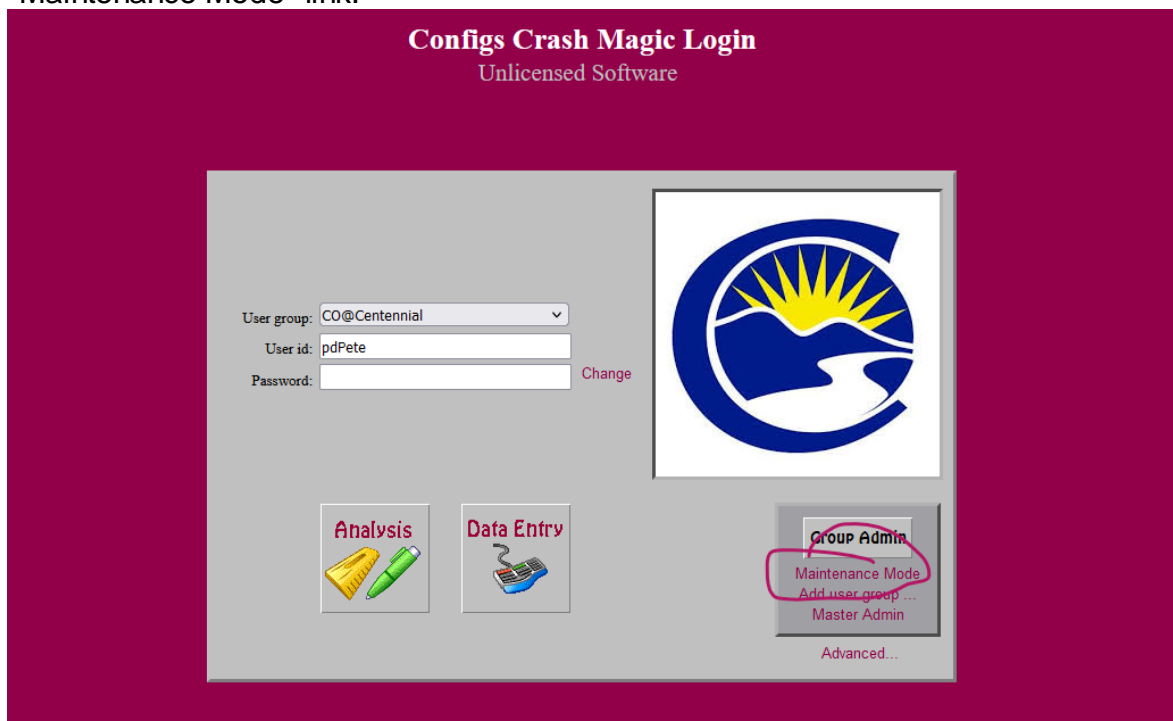
Be sure to review the Crash Magic requirements:

<https://www.pdmagic.com/kb/CurrentCrashMagicRequirements>

Before beginning, the database administrator should create a login to the database and schema/catalog where the Crash Magic system tables will be stored.

Access the Crash Magic Maintenance Form

1. Open Crash Magic in your browser. Select the "Advanced" link and then the "Maintenance Mode" link.



2. On the login page, enter the install admin login and password created during the initial installation process and click the Login button.

The Crash Magic Online Maintenance Form

The Crash Magic system is serving
and the system tables are ready. You
can log into the maintenance form
using either the install or .master login

Login:
Password:
Login

Ready - Wed 20:33:35 GMT

3. Click on the Settings tab of the Crash Magic Online Maintenance Form

Crash Magic Online Maintenance Form

Welcome | **Settings Wizard** | Advanced | DB Utilities | System Information |

Samples: Pd' SYS Legacy Connection Strings ----- Use sample as connection string

SYS Connection string: Provider=sqloledb;Data Source=cmDbServer\cmdev;Initial Catalog=cmSysConfigs;

SQL Server type: MSSQL2014

Cursor type: Static

Cursor location: Client

Login: cmSysConfigs_CM

Password:

Role:

Password:

Schema owner and prefix (i.e. dbo.Cm): dbo.cm

Fully qualified CID Table cmSysConfigs.dbo.cmCID

Valid admin account:

Master login:

Master password:

Navigate to: [https://dev.governmenttools.com/configs/cmo.dll/\\$/Start](https://dev.governmenttools.com/configs/cmo.dll/$/Start)

Ready - Wed 20:13:54 GMT

Create a connection string to the Crash Magic system tables location

1. Click on the Samples drop down menu under the configuration tab and select the database type that will be used(Pd Programming recommends selecting one of the

Pd' Default connection strings samples at the bottom of the list for your type of database)

2. Click the Use sample as connection string button to populate the SYS Connection string box

SYS Connection string: `Provider=SQLOLEDB.1;Persist Security Info=False;Initial Catalog=CmSysDe`

3. Change the connection string parameters to access your database(In most cases this will mean changing the Data Source=(Enter the name of your data source) and if required the Initial Catalog=(Enter the name of the catalog)

Examples

Oracle

Sample connection string provided:

```
Provider=OraOLEDB.Oracle;Data Source=MyOracleDB;
```

MyOracleDB should be changed to the data source specified in the Oracle tns names file. In the example below the CrashMagic is the specified database alias in the Oracle tns names file.

```
Provider=OraOLEDB.Oracle;Data Source=CrashMagic;
```

MS SQL Server

Sample connection string provided:

```
Provider=sqloledb;Data Source=myServerAddress;Initial Catalog=myDataBase;
```

myServerAddress would be changed to the name of the server hosting the database, and myDataBase would be changed to the name of the database used on the server. In the example below SQLDbServer is the name of the host and CrashMagic is the database on the server.

```
Provider=sqloledb;Data Source=SQLDbServer;Initial Catalog=CrashMagic;
```

DB2

Sample connection string provided

```
Provider=IBMDADB2;Database=myDataBase;Hostname=myServerAddress;Protocol=TCPIP; Port=50000;
```

myDatabase is the name of the DB2 database and Hostname is the name of the server where the database resides. In the example below CrashM is the database name and DB2Server is the name if the server where CrashM resides.

```
Provider=IBMDADB2;Database=CrashM;Hostname=DB2Server;Protocol=TCPIP; Port=50000;
```

Enter the information that Crash Magic will use to access the system tables

1. In the SQL Server type drop down menu select the type of database that will be used
2. Enter the login name to the database that Crash Magic will use to connect to the system tables that will be used(This login must be granted select, insert, update and delete privileges for the Crash Magic system tables)
3. Enter a password for the Login in the Password text box
4. In the Role text box enter any required role that will be required to access the Crash Magic system tables(This is optional as some database environments may choose not to use roles. This should be left blank if no role is in use.)
5. Enter any password required by the role(This should be left blank if no role is in use) and press the tab key to open the next section

6. In the following text box enter the schema name and table name prefix for the Crash Magic system tables in the text box using the . separator(The recommended table prefix name is cm Example dbo.cm where dbo is the name of the schema that the tables will reside, and the Crash Magic system tables will all start with cm in their names)
7. Press the tab key to move to the next section

Create the Crash Magic system Tables

1. Click the Create tables button

Crash Magic Online Maintenance Form

Welcome [Settings Wizard] [Advanced IDB Utilities] [System Information]

Create tables

Display SQL and then Execute SQL. The "access" and "namespaces" parameters are optional and will be applied to the new system tables if specified. The "login" and "role" values are used to create the new tables and must have the appropriate permissions.

Grant access to: Login:

Tables namespace (optional): Password:

Index namespace (optional): Role:

```
-- TABLE CHANGES FOR dbo.cmUser (29 changes):
-- REASON: New table required
CREATE TABLE dbo.cmUser (
  Name VARCHAR(255) DEFAULT 'empty' NOT NULL,
  Description VARCHAR(255),
  DateCreated DATETIME DEFAULT '1900-01-01 00:00:00.000' NOT NULL,
  DateModified DATETIME DEFAULT '1900-01-01 00:00:00.000' NOT NULL,
  DateAccessed DATETIME,
  ParentID INT,
  ViewState INT,
  CurTab VARCHAR(50),
  ObjType VARCHAR(50),
  IconLibrary VARCHAR(50),
  IconCatalog VARCHAR(50),
  IconName VARCHAR(50),
  LastVersion BIGINT,
  ExtMSize INT,
  XMLData VARCHAR(MAX),
  BlobData VARBINARY(MAX),
  UserGroupID INT DEFAULT -1 NOT NULL,
  UserID INT DEFAULT -1 NOT NULL,
  ULogin VARCHAR(50),
  UPassword VARCHAR(255),
  CanAdmin INT,
  CurAdminUserGroupID INT,
  CurAdminUserID INT,
  CurAdminPRAttrID INT,
  CurUserGroupID INT,
  CurUserID INT,
```

Done

Navigate to: <https://dev.pdmagic.com/alpha/cma.dll/8/Start>

Ready - Tue 11:33:47 GMT

2. Ensure that the Grant access to box contains the user login name that Crash Magic will use to access the system tables.
3. In the Tables namespace box enter a table space name where the tables will be stored(This box should be left blank to use the default table space for the database)
4. In the Index namespace enter the table space name where the indexes for the system tables will be stored(This box should be left blank to use the default table space for the database)
5. In the Login text box enter a login name for a user that has the privilege to create tables
6. In the Password text box enter a password the login on the previous step
7. If required enter a role name in the Role text box(This should be left blank if the login requires no role to create tables)
8. If required enter a password for the role in the Password text box(If no role is used this box should be left blank)
9. Click the Display SQL button to render the script that will create the system tables and review the SQL
10. Click the Execute SQL button to create the system tables

11. Click the OK button when the SQL is done executing
12. Click the Done button to close the window
13. Click the Welcome tab of the Crash Magic Online Maintenance Form to save your changes and close the browser window.

Prepare connection to crash data

Create a connection string to the Crash Data

1. Locate the crash database and prepare a connection string to connect to it.
2. Provide this connection string, login and password information to the Group Administrator to allow Crash Magic to access the collision data. This login must have select access to the collision data (Clients that also use data entry within Crash Magic will need a login and password that has select, update and insert privileges to the collision tables)

Inform the Group Administrator for Crash Magic that they may proceed with the [Install - Group administrator steps](#)¹⁷² in this manual.

4. Group administrator steps

Once the database administrator has prepared the tables and permissions for use by Crash Magic, and the server administrator has installed the program and checked that it responds on the network, it is time for the group administrator to finish the installation process.

Configure Crash Magic

This process will update the information Crash Magic stores in the database, and allow the program to read the collision data. The xml configuration file should be available on the computer that the Group Administrator is using along with access to the license information provided by Pd' Programming.

Log into the Crash Magic web site

1. Open the Crash Magic web site using the URL provide by the server administrator
2. Enter the login and password created by the server administrator to administer Crash Magic
3. Click the login button to enter the Crash Magic Online Maintenance Form

Enter the license information

1. While on the "Welcome" tab of the Crash Magic Online Maintenance enter the Form Click on the Licensed to box enter the name Crash Magic has been licensed to
2. Click on the ID field and enter the ID that was provided by Pd' Programming with the license information

3. Click on the License Key field and enter the license key(It is recommended that you copy and paste this information)

Create a Master Administrator login for the instance

1. While logged into the Crash Magic Online Maintenance Form click on the Configuration Wizard tab
2. Ensure that the Create Master Administrator account button is visible(If this is not visible Crash Magic is not able to connect to the system tables) If the screen shows a prompt for Master login: and Master password: than one has already been created.
3. Click the Create Master Administrator account button
4. Enter a desired name that will be used to login for the Master Admin(MasterAdmin is the recommended name. This login should be retained in safe place for support purposes)
5. Enter a desired password for the Master Admin login in the Master Admin Password box
6. Repeat the password in the Repeat password box
7. Click the Create Master Admin button
8. Click OK when the master user has been added
9. Click the Done button to close the window

Update the program resources

1. While logged into the maintenance form, select the "Welcome" tab. Allow the page to update.
2. If program resources need updating, the screen show the available updates and will provide a button to start the process.Click that button.
3. This process may take a few minutes. The progress bar will update during this time)
4. If the process completes without error, the link to the home page at the bottom of the screen will become available.

Start the Crash Magic application

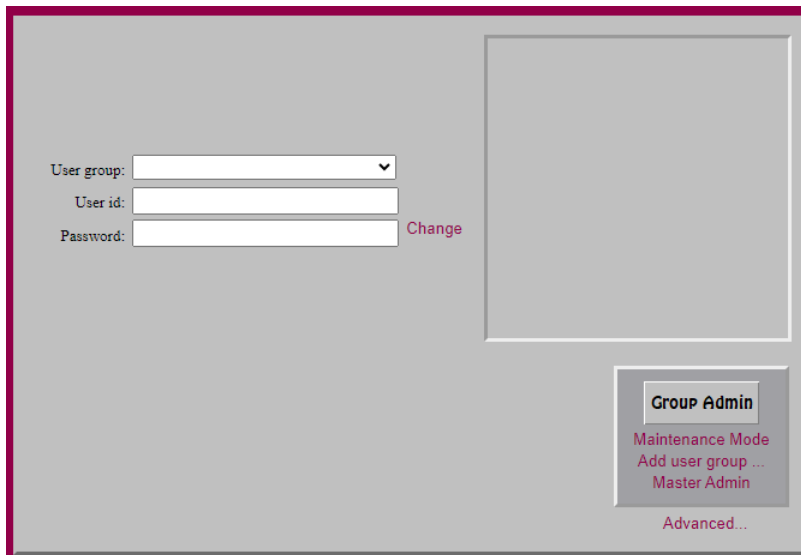
1. While still on the "Welcome" tab of the Crash Magic Online Maintenance Form, click the Start serving button to start Crash Magic
2. Click the Navigate to button to open the Crash Magic login window

Import configuration(s)

Start by verifying access to the login page. The project manager should be able to provide the URL.

Because this is a brand new installation, with no configuration installed, the logo area will be empty or will contain the Pd' Magic logo. There will also be no user groups

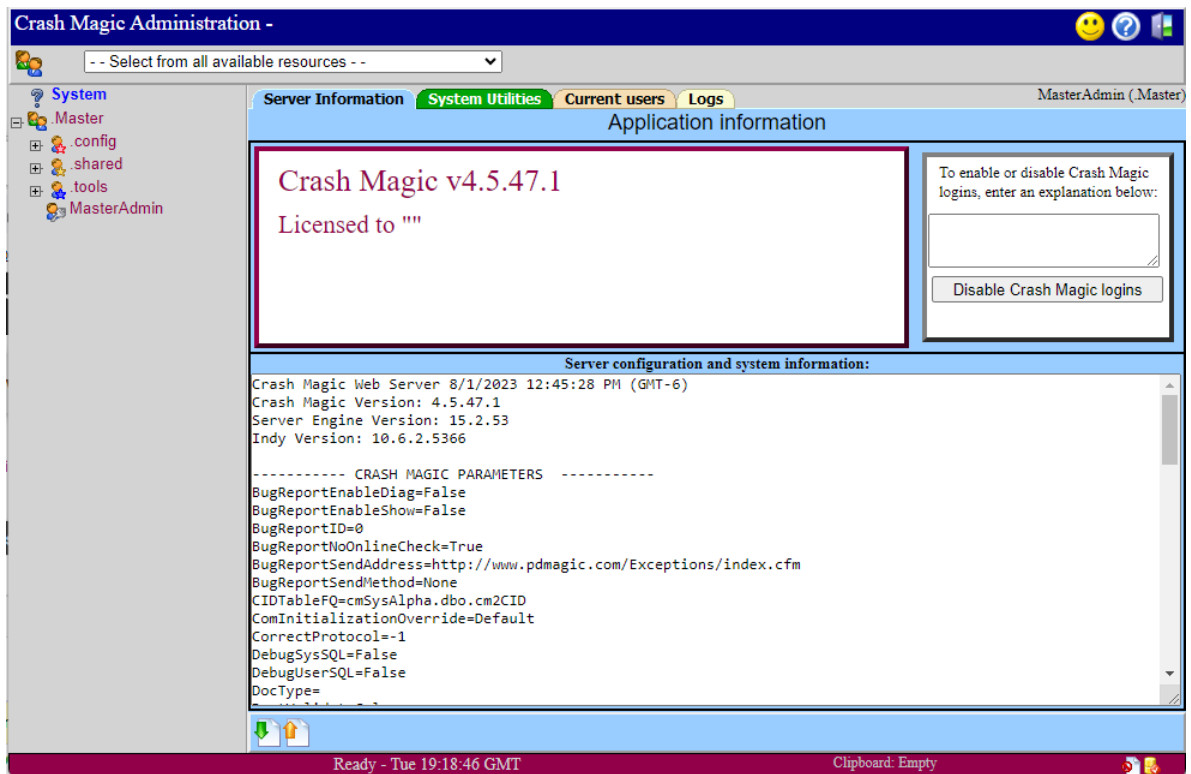
available. The Master Admin login, created in the previous step, must be used to import the provided user groups containing the client-specific configuration data.



1. Enter the Master Admin login and password.
2. Select the "Advanced..." link
3. Select the "Master Admin" link

Important. Clients should only utilize the Master Admin login once and only during this initial installation phase. After the configuration files have been imported, the Master Admin password should not be utilized again without guidance from Pd' Programming. Pd' Programming can not be responsible for problems resulting from Master Admin access after the initial installation.

Crash Magic configurations are stored as user groups. Upon login, there will be only a single user group. The .Master user group will be shown on the left side of the screen. It contains resources that are common to all Crash Magic configurations.

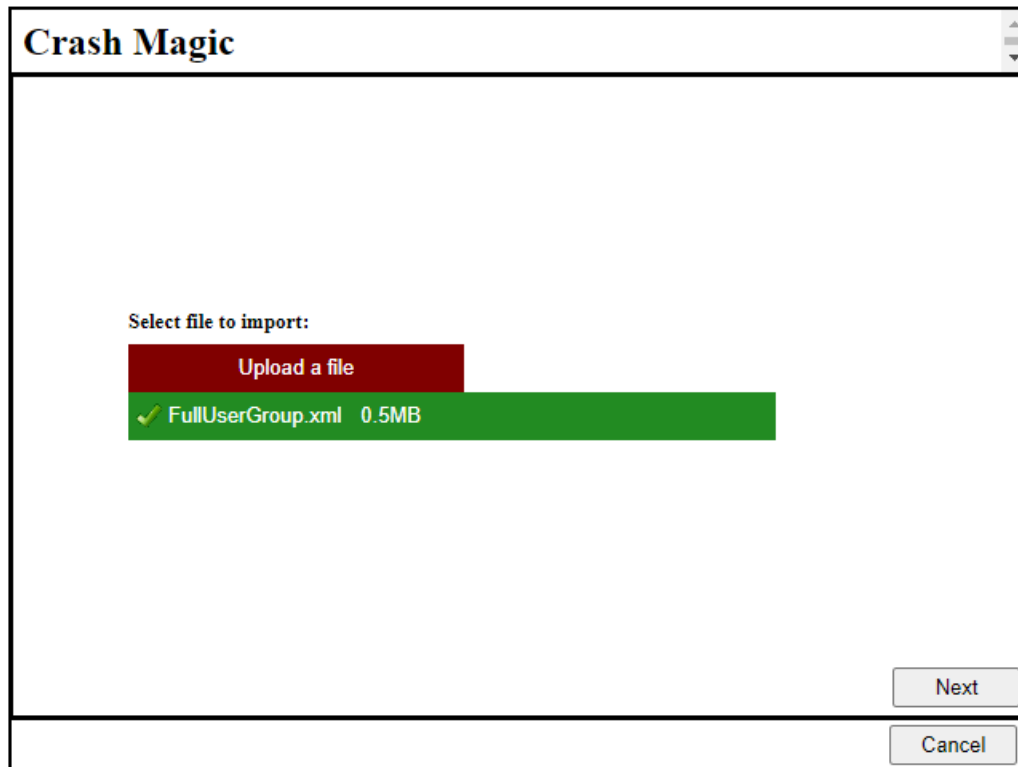


Depending on the structure of your configuration, there will be one or more configuration groups to import. There will always be a "ConfigUser" and "SharedUser" for your agency. The user group name will be <State>@<Agency> such as "CO@DOT" or "[CO@Boulder](#)". In addition, there may be several "shared" groups. Shared groups are prefixed with an underscore "_" such as "_CO@BaseConfig".

For each of the provided XML configuration files:

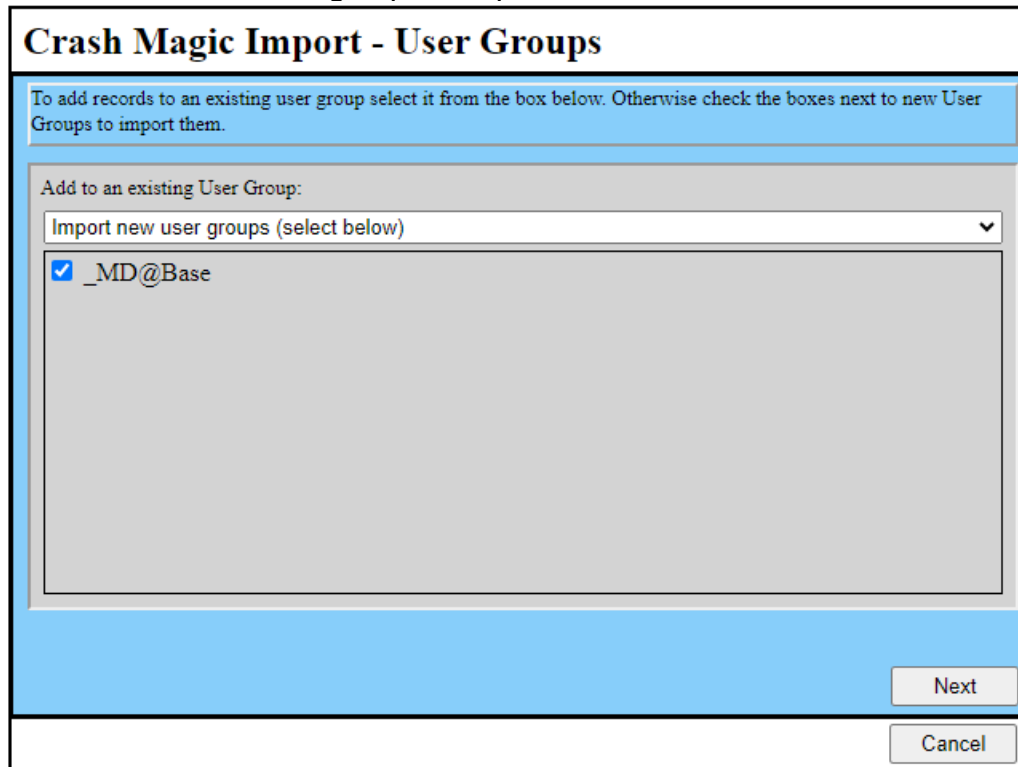
1. Choose the orange import wizard button at the bottom of the screen.

2. Select "Upload a file" and choose the XML file. Click the "Next" button.



The screenshot shows a window titled "Crash Magic". Inside, the text "Select file to import:" is displayed. Below it, there are two buttons: a red "Upload a file" button and a green button labeled "✓ FullUserGroup.xml 0.5MB". At the bottom right, there are "Next" and "Cancel" buttons.

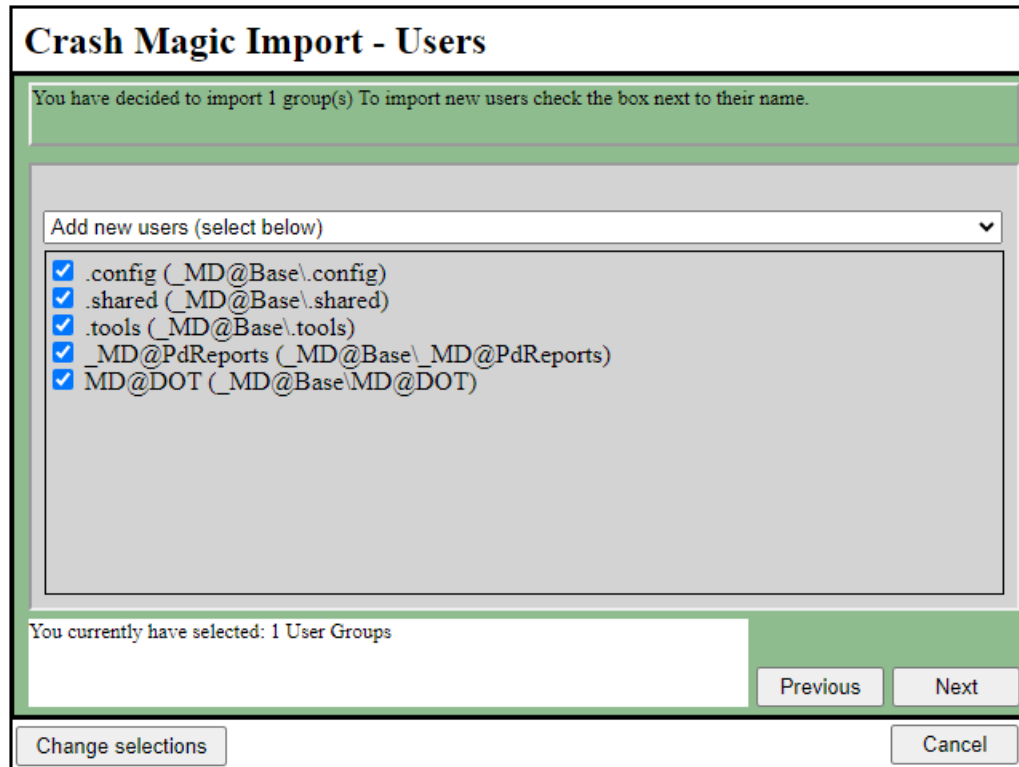
3. Select the default "user group" to import and select "Next".



The screenshot shows a window titled "Crash Magic Import - User Groups". It contains a blue header bar with the text: "To add records to an existing user group select it from the box below. Otherwise check the boxes next to new User Groups to import them." Below this, there is a section "Add to an existing User Group:" with a dropdown menu showing "Import new user groups (select below)". Under the dropdown, there is a list with a checked checkbox next to "_MD@Base". At the bottom right, there are "Next" and "Cancel" buttons.

4. Select the desired "users". Usually this will include a ".config", ".shared", ".tools" and any references to other user groups that should have access to this one. Generally,

select all of the listed users. Select "Next".



Crash Magic Import - Users

You have decided to import 1 group(s) To import new users check the box next to their name.

Add new users (select below) ▼

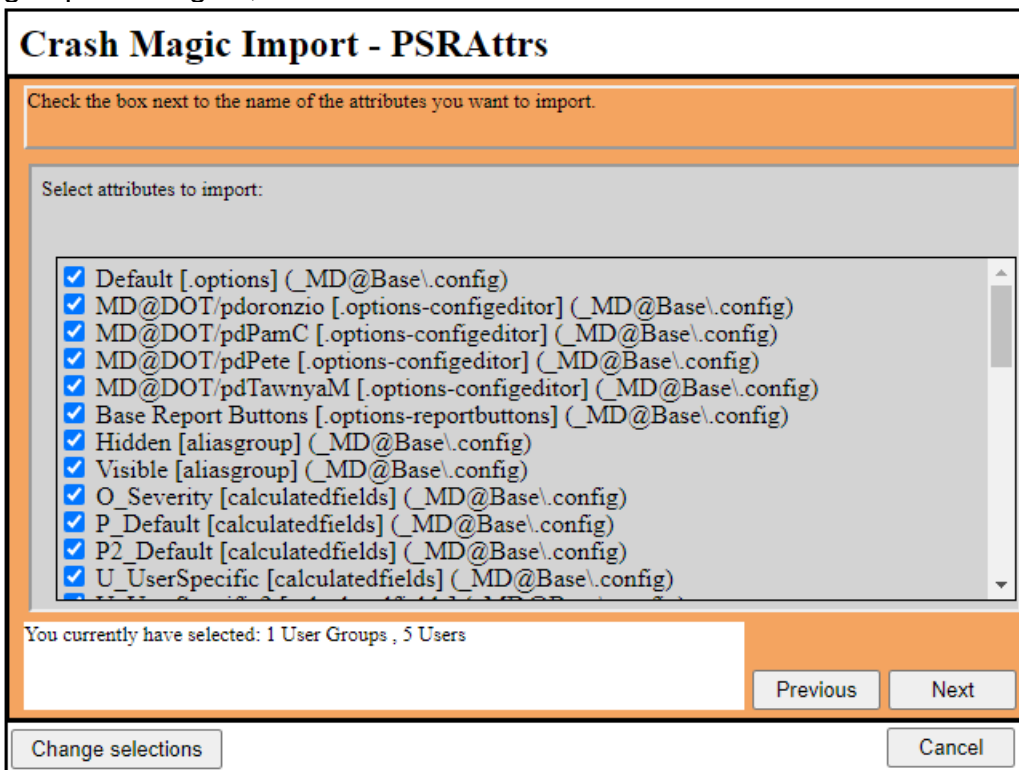
- ☒ .config (_MD@Base\.config)
- ☒ .shared (_MD@Base\.shared)
- ☒ .tools (_MD@Base\.tools)
- ☒ _MD@PdReports (_MD@Base_MD@PdReports)
- ☒ MD@DOT (_MD@Base\MD@DOT)

You currently have selected: 1 User Groups

Previous Next

Change selections Cancel

5. The next screen lists all of the available resources (PSRattrs) in the specified group/user. Again, select all of them and click "Next".



Crash Magic Import - PSRAttrs

Check the box next to the name of the attributes you want to import.

Select attributes to import:

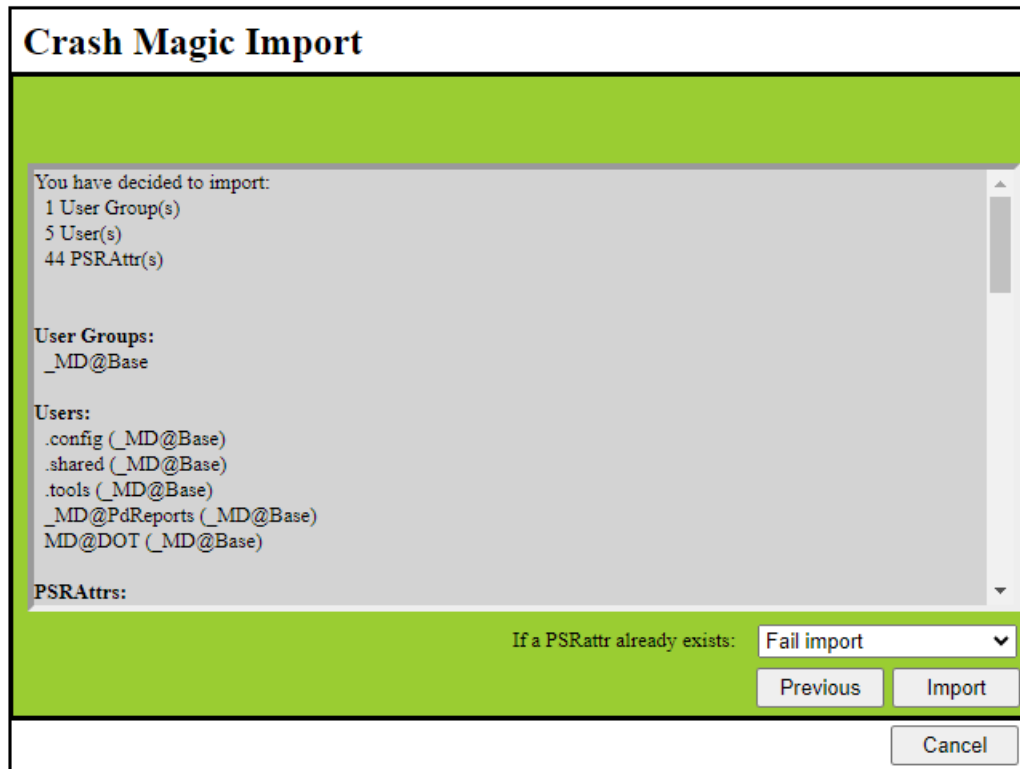
- ☒ Default [.options] (_MD@Base\.config)
- ☒ MD@DOT/pdoronzio [.options-configeditor] (_MD@Base\.config)
- ☒ MD@DOT/pdPamC [.options-configeditor] (_MD@Base\.config)
- ☒ MD@DOT/pdPete [.options-configeditor] (_MD@Base\.config)
- ☒ MD@DOT/pdTawnyaM [.options-configeditor] (_MD@Base\.config)
- ☒ Base Report Buttons [.options-reportbuttons] (_MD@Base\.config)
- ☒ Hidden [aliasgroup] (_MD@Base\.config)
- ☒ Visible [aliasgroup] (_MD@Base\.config)
- ☒ O_Severity [calculatedfields] (_MD@Base\.config)
- ☒ P_Default [calculatedfields] (_MD@Base\.config)
- ☒ P2_Default [calculatedfields] (_MD@Base\.config)
- ☒ U_UserSpecific [calculatedfields] (_MD@Base\.config)

You currently have selected: 1 User Groups , 5 Users

Previous Next

Change selections Cancel

6. Select "Next" on the next three screens that should be empty. (Projects, Studies and Reports)
7. The final screen summarizes your selections. Simply select "Import". When the process is complete, select "Done" to close the screen and return to the administration screen.



Crash Magic Import

You have decided to import:

- 1 User Group(s)
- 5 User(s)
- 44 PSRAttr(s)

User Groups:

- _MD@Base

Users:

- .config (_MD@Base)
- .shared (_MD@Base)
- .tools (_MD@Base)
- _MD@PdReports (_MD@Base)
- MD@DOT (_MD@Base)

PSRAttrs:

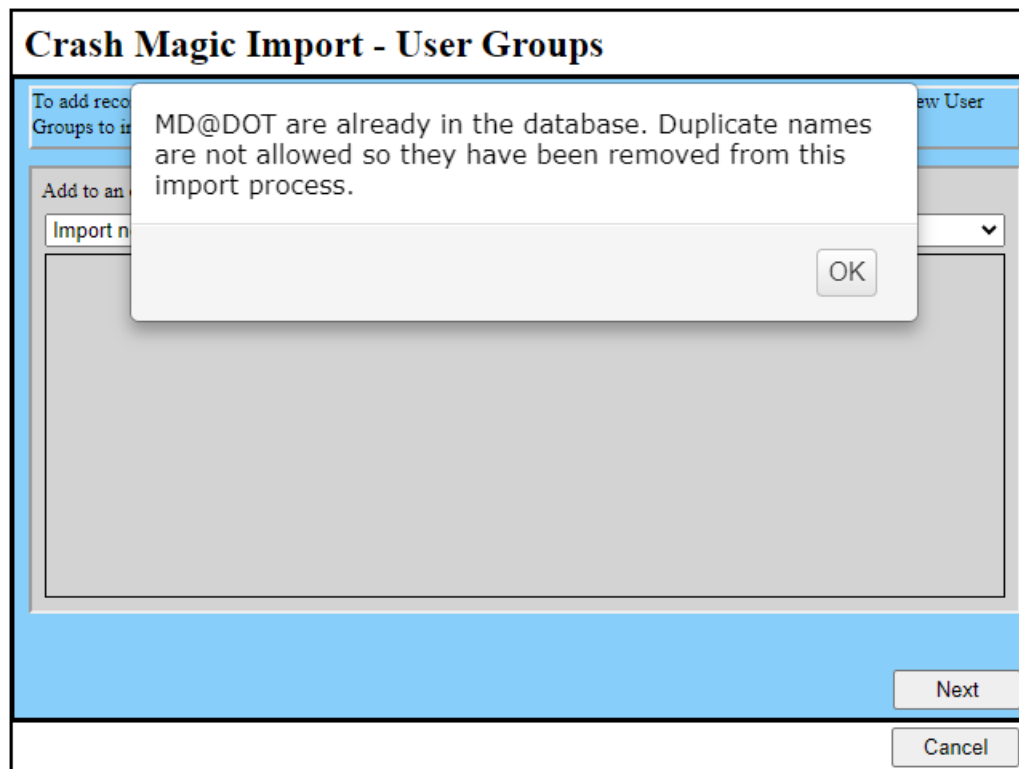
If a PSRattr already exists: Fail import

Previous Import

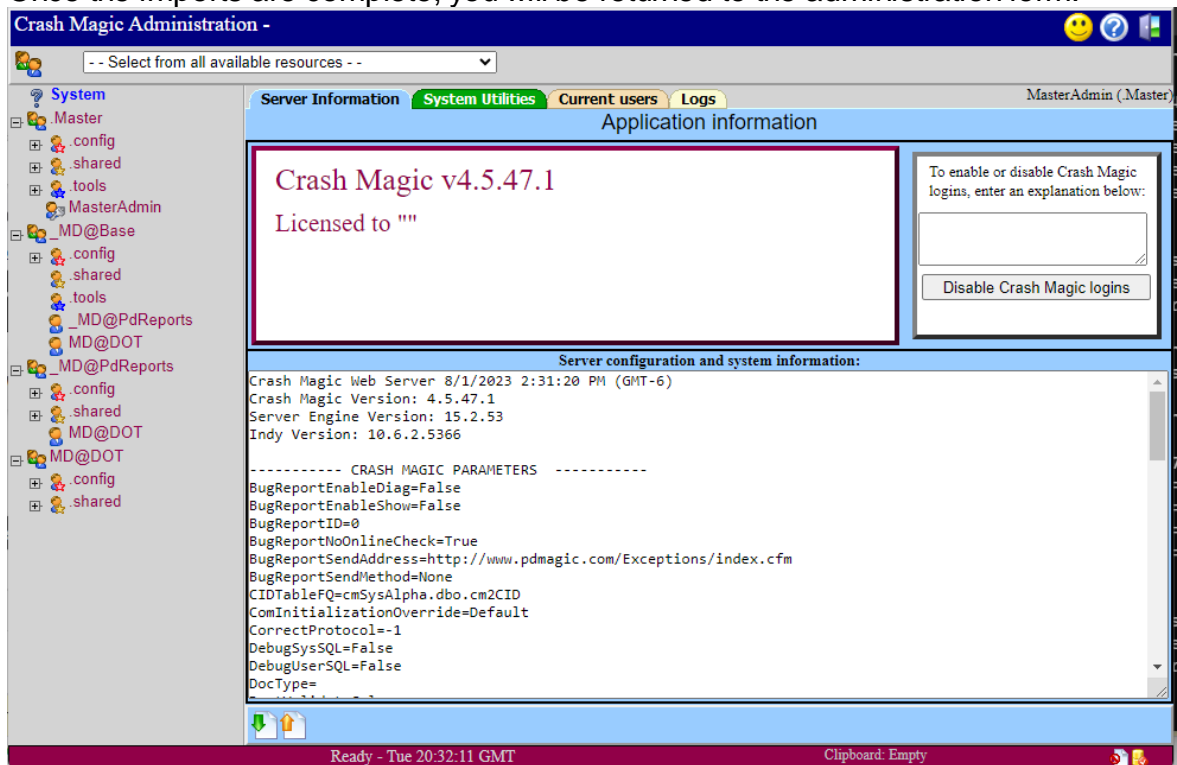
Cancel

8. Repeat this process for each of the provided XML configuration files.
Note: Some of the XML files that you import will contain all of the users for the group. Others will just contain a single user. In the latter case, you will be informed that the user group is already in the database. It will not import that group again and will instead present a message indicating that you must select a group to install into.

Simply select the appropriate group and continue the import process as usual.




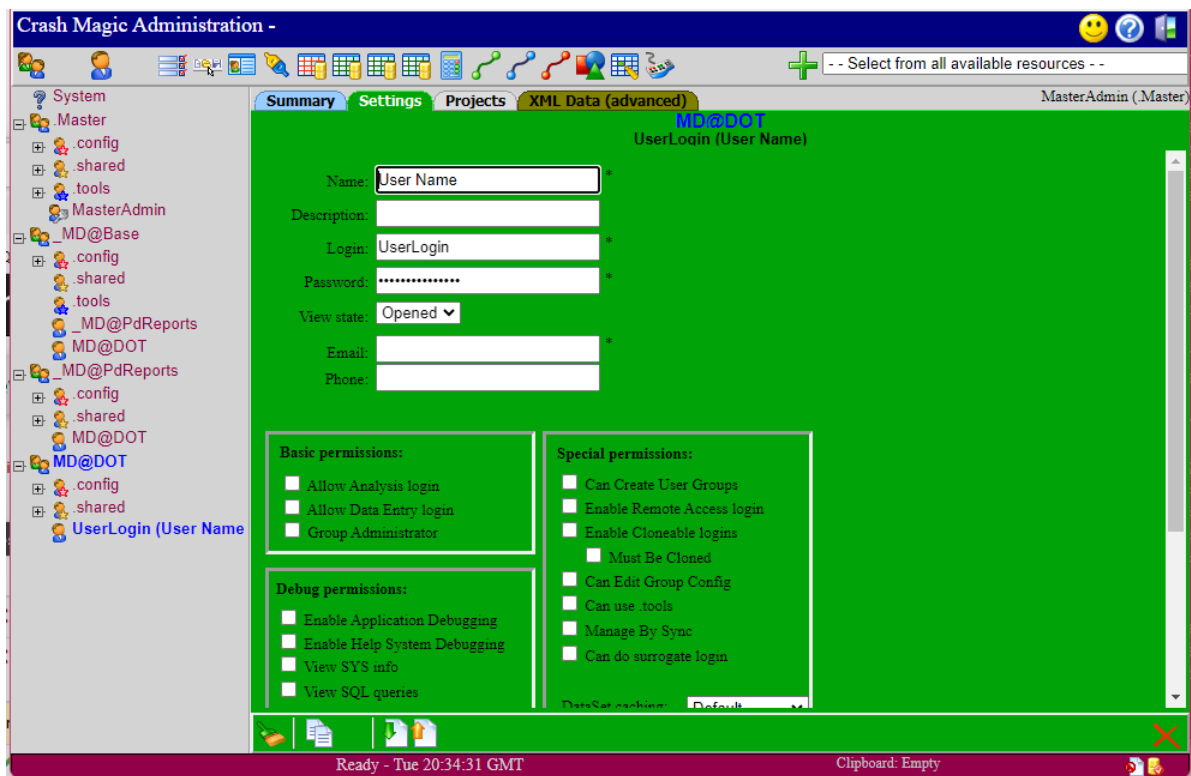
9. Once the imports are complete, you will be returned to the administration form.



Add an administrative user

Once logged into Crash Magic as a group administrator new users can be added to Crash Magic:

1. Click on the configuration (agency) name located in the tree on left of the window to ensure the user is created in the correct group
2. Click the  button above the tree to create a new user
3. Click on the green Settings tab
4. Enter the name of the user in the Name field
5. Enter the login name in the Login field
6. Type in a password for the login
7. Enter the users email and (optionally) phone number
8. Check the Allow Analysis login and Group Administrator check boxes



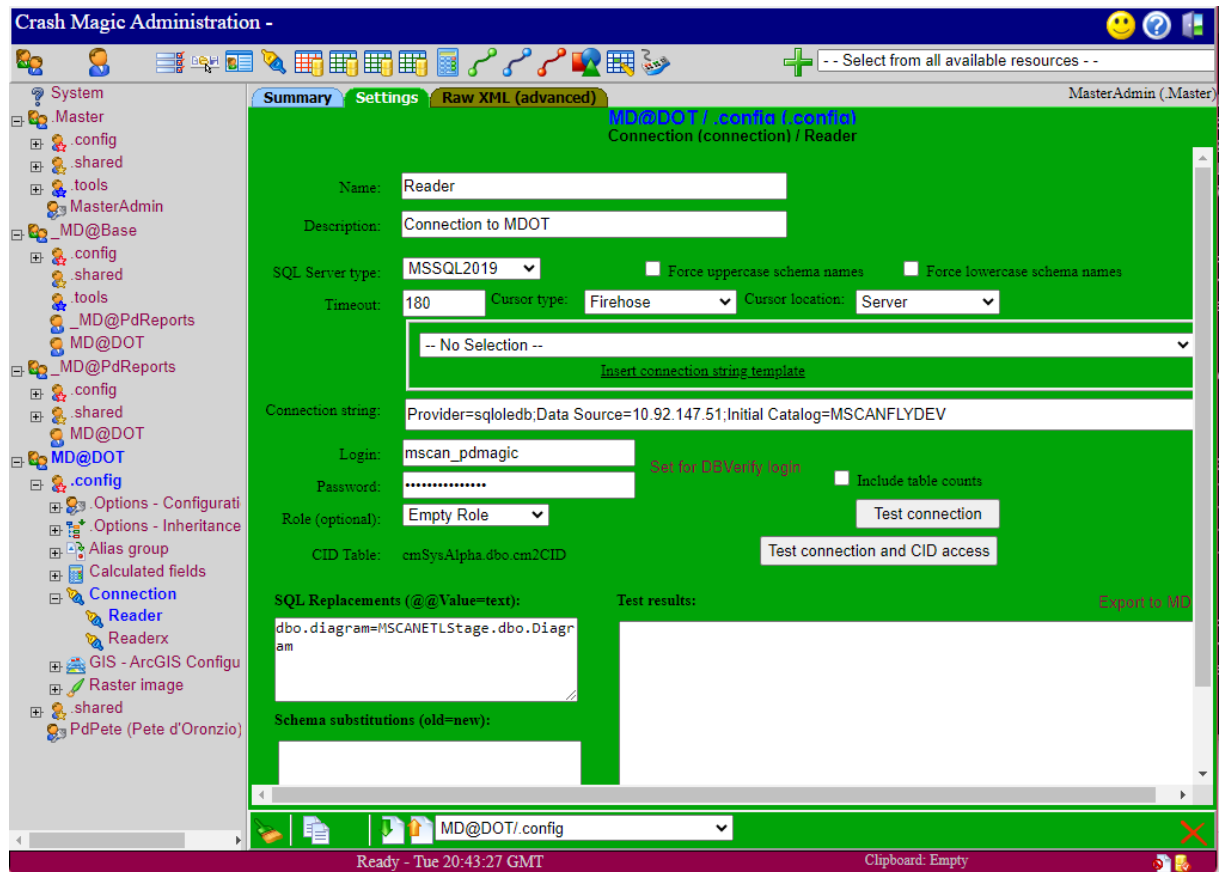
Crash Magic will automatically save the user information as you move to the next step.

Update the crash database connection information

Clients that have purchased a configuration will need to update the connections to point to their internal collision database. In general Pd' Programming creates configurations based on a sample copy of the clients collision database. Configurations sent to users therefore contain database connection information for sample database at Pd' Programming, and must be changed. Each configuration can contain one or more connections to the collision database. The information required by these connections

should be provided by the database administrator of the collision data. Here is how to locate the connections:

1. Log into Crash Magic as the Group Admin created in the prior step.
2. In the admin tree on the left, click on the plus sign (+) next to the .config user under the user group that your are currently logged into.
3. Click the plus sign (+) next to the connection name in the admin tree. This will provide access to the available connections.



In this example two connections are available for the MD@DOT user group. A Reader and a Readerx connection are shown in this example.

4. Select a connection to update by clicking on it in the tree (For the next examples we will select the Reader connection).
5. Click on the green settings tab. You will receive a warning indicating that this resource is not available to you. That's okay. It's because you're logged in as "Master Admin".
6. Change the Data Source of the connection string to the data source for the collision data (For MS SQL this will be the name of your database server. For Oracle this will be the name of the database service specified in the tnsnames file. With DB2 this is name used by the client for the database).
7. If your connection string includes a catalog change the catalog name to the name of the database (This is generally only for MS SQL).

8. Once the connection string is changed, update the login and password.
9. Click the "Test connection" button to verify that each connection works.
Note: Some databases use roles to manage permissions on the database. A default empty role is included in the configuration. Roles that are already defined will be displayed in the Role drop down list. If your logins require roles on the database you may need to add or change the roles provided. See the [Roles](#) ²⁷⁹ section of the manual.

At this point, the installation and setup of Crash Magic is complete.

Log out of Crash Magic. Do not perform any additional work as the Master Admin. All future work must be completed using the new administrative account.

5.4 Reference

Data Entry

Data entry defines the forms used by the data entry portion of the program. The dataentrydefinition defines the form and validationrules are used to validate the data before the record is inserted into the database.

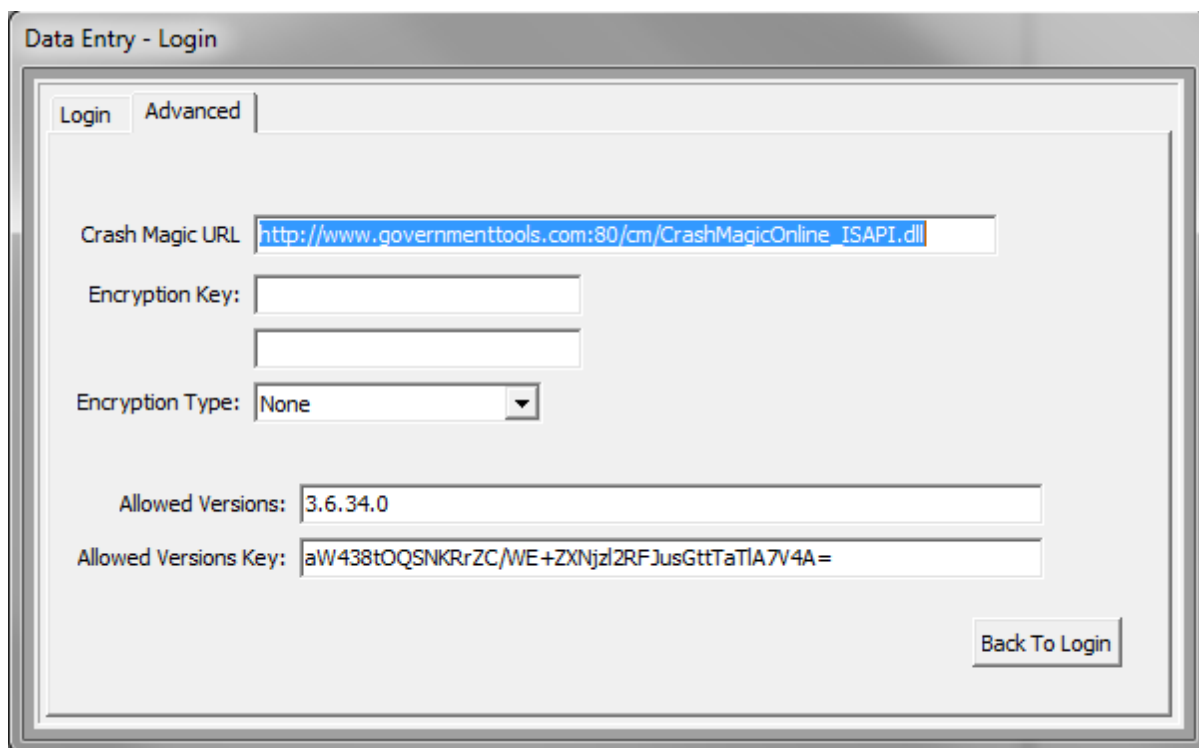
Data Entry Versions

The Data Entry program is tied to the version of Crash Magic that is running. The version of Crash Magic must match the version of Data Entry that is being used or the user will receive an error.

Newer versions of Data Entry can be run against older versions of Crash Magic with a key from Pd' Programming.

To Enable a newer version of data entry with an older version of Crash Magic:

1. Open the data entry program.
2. Click OK on the error messages that notify the user that the version of Crash Magic does not match data entry.
3. Click the advanced tab.
4. Enter the Crash Magic web site URL in the Crash Magic URL box.
5. Enter the version of Crash Magic that will be used.
6. Enter the Allowed Version Key in the Allowed Version Key text box.



Data Entry - Login

Login | Advanced

Crash Magic URL:

Encryption Key:

Encryption Type:

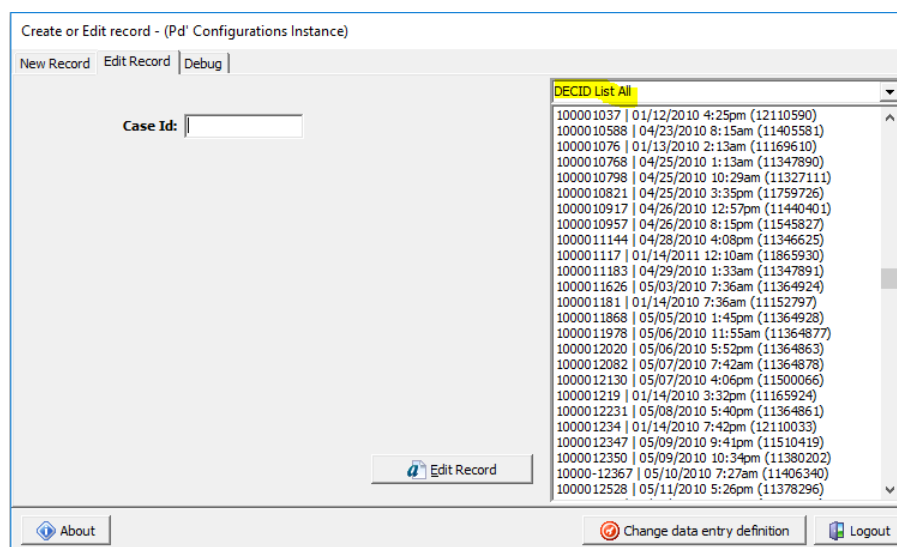
Allowed Versions:

Allowed Versions Key:

[Back To Login](#)

Caseld Listing

The Crash Magic Data Entry application is capable of providing a selection list of crashes for editing. In order for this list to appear the data entry definition must be configured for it. The list only shows the top 500 items.



Create or Edit record - (Pd' Configurations Instance)

New Record | Edit Record | Debug

Case Id:

[Edit Record](#)

DECID List All

100001037	01/12/2010 4:25pm (12110590)
1000010588	04/23/2010 8:15am (11405581)
100001076	01/13/2010 2:13am (11169610)
1000010768	04/25/2010 1:13am (11347890)
1000010798	04/25/2010 10:29am (11327111)
1000010821	04/25/2010 3:35pm (11759726)
1000010917	04/26/2010 12:57pm (11440401)
1000010957	04/26/2010 8:15pm (11545827)
1000011144	04/28/2010 4:08pm (11346625)
100001117	01/14/2011 12:10am (11865930)
1000011183	04/29/2010 1:33am (11347891)
1000011626	05/03/2010 7:36am (11364924)
100001181	01/14/2010 7:36am (11152797)
1000011868	05/05/2010 1:45pm (11364928)
1000011978	05/06/2010 11:55am (11364877)
1000012020	05/06/2010 5:52pm (11364863)
1000012082	05/07/2010 7:42am (11364878)
1000012130	05/07/2010 4:06pm (11500066)
100001219	01/14/2010 3:32pm (11165924)
1000012231	05/08/2010 5:40pm (11364861)
100001234	01/14/2010 7:42pm (12110033)
1000012347	05/09/2010 9:41pm (11510419)
1000012350	05/09/2010 10:34pm (11380202)
10000-12367	05/10/2010 7:27am (11406340)
1000012528	05/11/2010 5:26pm (11378296)

[About](#) [Change data entry definition](#) [Logout](#)

Sample Case Id list

In order to enable display of this list, the following steps must be taken:

1. You must already have a working data entry definition.
2. Log into Crash Magic as Analysis.
3. Create a new field list called "DECrashes"
 - a. Select an existing study or create a new study
 - b. Create a new "Listing" report. (e.g. The Caseld template will work)
 - c. Create a new Field List for this report
 - d. Edit the new field list so that it contains exactly two fields. The title for those fields must be "Caseld" and "Description".
 - i. Caseld - This value must match the Caseld being prompted for in the data entry definition.
 - ii. Description - This value may contain any desired text. Common values are date, time, other id fields, etc. Multiple fields may be used by editing the "expression" for that field.
 - e. Note the names of the database fields that you use to create the Caseld and Description. You will need to know these later.
 - f. Rename the new Field List "DECrashes". (or any other name. We'll use this name later)
 - g. Accept the changes and view the listing report - verify that it contains two fields called Caseld and Description, and that the values are as you intended.
4. Log in to Crash Magic as Administrator
5. Create a "Query - General" called "DECrashes All".
 - a. Make sure that it contains all of the fields that you used in the DECrashes field list. This query will be used to populate the field list you created.
 - b. Name the query "DECrashes All". You may use another name, but it must start with the same name as the field list that you created.
 - c. If desired, create additional queries with the same fields, but with a WHERE clause limiting the list to crashes in need of attention. (e.g. WHERE TypeOfCollision IS NULL) Name these queries with the same prefix. (e.g. DECrashes - TOC)
6. Update your Data Entry Definition so that it will display case numbers when preparing to create or edit a crash.
 - a. Load the desired data entry definition
 - b. Select the "UNIQUE FIELDS" check box for the Main crash record. It will likely only contain a Caseld entry field. That field will have a control type of "String".
 - c. Add an item to the "Properties" box called QueryPrefix. It's value must be the same as the field list that you created, and therefore be a prefix to the query that you created. For this example, it will be: QueryPrefix=DECrashes
7. Log out of Crash Magic
8. Test the data entry definition by running the data entry application

Database

Database Conventions

The following sections describe database conventions and system tables for the Crash Magic database.

Crash Magic uses several tables to store internal and user data for the program. These tables can be created through the Crash Magic Configuration Editor or created manually with the script produced by the Editor. Each table name must start with the prefix listed in the SQLTblPrefix box of the Crash Magic Configuration Editor for the configuration in use. Table prefix names should be limited to less than four characters.

The following is an example of the tables used by Crash Magic with a table prefix of cm1:

- cm1ALIASES
- cm1CID
- cm1UGROUP
- cm1LOGGEDIN
- cm1PROJECT
- cm1PSRATTR
- cm1REPORT
- cm1STUDY
- cm1USER

Crash Magic system tables

Crash Magic uses system table to store user configuration and user data. Tables and fields are specific to the version of Crash Magic in use, and may not match the tables shown.

Note - the XML data is stored in CLOB fields for Oracle as shown in the following list. Prior to Oracle 9i the XML fields were Long data type. In MS SQL server the data type of these fields are memo.

Aliases table (ALIASES) - Contains location alias information

- UserGroupID - Number field- The user group of the alias
- UserID - Number field- The user of the alias
- AliasID - Number field- The id of the alias
- PSRAttrID - Number field- The id number of the PSRattr
- DateCreated - Date field- Date user record was created
- DateModified - Date field- The last date the user record was modified
- DateAccessed - Date field- Last date the record was accessed for archive purposes
- LocType - Number field- The type of location the alias record will be used for
- FromRule1 - (50) Character field- Location information

- FromRule1Unique1 - (50) Character field- Unique rule condition
- FromRule1Unique2 - (50) Character field- Unique rule condition
- ToRule1 - (50) Character field- Alias location information
- ToRule1Unique1 - (50) Character field- Unique rule condition
- ToRule1Unique2 - (50) Character field- Unique rule condition
- FromRule2 - (50) Character field- Location information
- FromRule2Unique1 - (50) Character field- Unique rule condition
- FromRule2Unique2 - (50) Character field- Unique rule condition
- ToRule2 - (50) Character field- Alias location information
- ToRule2Unique1 - (50) Character field- Unique rule condition
- ToRule2Unique2 - (50) Character field- Unique rule condition

CaseID Table (CID) - Contains crash record IDs for use with GIS and third party application calls

- UserID - Number field- The user ID of the record
- ProjectID - Number field- The project id of the record
- StudyID - Number field- The study id of the record
- CIDID - Number field- The crash id.
- DateCreated - Date field- Date user record was created.
- DateModified - Date field- The last date the user record was modified
- DateAccessed - Date field- The last date the user record was accessed for archive purposes
- UniqueStr - (50) Character field- Unique string value for the record
- UniqueDate - Date field- Unique date value for the record
- UniqueNum1 - Number field- Unique ID for the record
- UniqueNum2 - Number field- Unique ID for the record
- UniqueNum3 - Number field- Unique ID for the record
- Name - (50) Character field- Record name

Group Table (UGROUP)- Contains Information on Crash Magic user groups

- Name -(50) Character field- Name of group
- Description -(255) Character field - Description of the group
- DateCreated -Date field- Date group record was created
- DateModified -Date field- The last date the record was modified
- DateAccessed -Date field- Last date the record was accessed for archive purposes
- ParentID -Number field- Contains the parent of record
- ViewState -Number field- Stores the view state code for the current group
- CurTab -(50) Character field- The current tab the group record is displaying
- ObjType -(50) Character field- The type of object the program will store the in the record

- EstXMLSize -Number field- Size of the XML data stored in the XMLData field
- XMLData -CLOB field- Specific XML data for the group record
- UserGroupID -Number field- The group that the user record belongs to
- SharedUserID -Number field- The user id that contains the default objects for the group

Logged In Table Table (LOGGEDIN)- Contains Information on who is logged into Crash Magic

Name -(50) Character field- User id of the person logged in

Description -(255) Character field - Description of the user logged in

DateCreated -Date field- Date group record was created

DateModified -Date field- The last date the record was modified

DateAccessed -Date field- Last date the record was accessed for archive purposes

ParentID -Number field- Contains the parent of record

ViewState -Number field- Stores the view state code for the current logged in user

CurTab -(50) Character field- The current tab the user is displaying

ObjType -(50) Character field- The type of object the program will store the in the record

EstXMLSize -Number field- Size of the XML data stored in the XMLData field

XMLData -CLOB field- Specific XML data for the logged in user record

UserGroupID -Number field- The group that the logged in user record belongs to

UserID -Number field- The user ID of the user that is logged in

LoggedInID -Number field- ID assigned to the logged in user

MachineName -(1000) Character field- Name of the machine in use by the user

SessionId -(50) Character field- Session ID of the user

Project Table (PROJECT)- Stores information on Crash Magic projects

- Name -(50) Character field- Name of project
- Description -(255) Character field - Description of the project
- DateCreated -Date field- Date the project record was created
- DateModified -Date field- The last date the project was modified
- DateAccessed -Date field- Last date the record was accessed for archive purposes
- ParentID -Number field- Contains the parent of the project
- ViewState -Number field- Stores the view state code for the current project
- CurTab -(50) Character field- The current tab the project is displaying
- ObjType -(50) Character field- The type of object the program will store the record in
- EstXMLSize -Number field- The size of the project XML data
- XMLData -CLOB field- Specific XML data for the project record
- UserGroupID -Number field- The group that the user record belongs to
- UserID -Number field- The user id that owns the project
- ProjectID -Number field- A number to identify the current project

PSRattr Table (PSRATTR)- Table to store project study and report attributes

- Name -(50) Character field- Name of the PSRattr record
- Description -(255) Character field - Description of the PSRattr
- DateCreated -Date field- Date PSRattr record was created
- DateModified -Date field- The last date the PSRattr record was modified
- DateAccessed -Date field- Last date the record was accessed for archive purposes
- ParentID -Number field- Contains the parent of the PSRattr
- ViewState -Number field- Stores the view state code for the current PSRattr
- CurTab -(50) Character field- The current tab the PSRattr record is displaying
- ObjType -(50) Character field- The type of object the program will store the record in
- EstXMLSize -Number field- Size of the XML data stored in the XMLData field
- XMLData -CLOB field- Specific XML data for the PSRattr record
- UserGroupID -Number field- The group that the PSRattr record belongs to
- UserID -Number field- The user ID the PSRattr record belongs to
- PSRattrID -Number field- The ID number of the PSRattr
- ProjectID -Number field- The ID of the project under which the PSRattr resides
- StudyID -Number field- The ID of the study under which the PSRattr resides
- ReportID -Number field- The ID of the study under which the PSRattr resides

Report Table (REPORT)- Stores information on Crash Magic reports

- Name -(50) Character field- Name of the report
- Description -(255) Character field - Description of the report
- DateCreated -Date field- Date report was created
- DateModified -Date field- The last date the report was modified
- DateAccessed -Date field- Last date the record was accessed for archive purposes
- ParentID -Number field- Contains the parent of record of the report
- ViewState -Number field- Stores the view state code for the current report
- CurTab -(50) Character field- The current tab the report record is displaying
- ObjType -(50) Character field- The type of object the program will store the record in
- EstXMLSize -Number field- The estimated size of the XML data stored in the record
- XMLData -CLOB field- Specific XML data for the report record
- UserGroupID -Number field- The group that the report belongs to
- UserID -Number field- The user id that the report belongs to
- ReportID -Number field- The id to identify the report
- StudyID -Number field- The ID that identifies study the report belongs to
- ProjectID -Number field- The project that report belongs to

Study Table (STUDY)- Stores information on Crash Magic studies

- Name -(50) Character field- Name of the study
- Description -(255) Character field - Description of the study
- DateCreated -Date field- Date study was created

- DateModified -Date field- The last date the study was modified
- DateAccessed -Date field- Last date the record was accessed for archive purposes
- ParentID -Number field- Contains the parent of record of the study
- ViewState -Number field- Stores the view state code for the study
- CurTab -(50) Character field- The current tab the study record is displaying
- ObjType -(50) Character field- The type of object the program will store the record in
- EstXMLSize -Number field- The size of the XML data in the study
- XMLData -CLOB field- Specific XML data for the study record
- UserGroupID -Number field- The group that the user record belongs to
- UserID -Number field- The user id that owns the study
- StudyID -Number field- The id that identifies the study
- ProjectID -Number field- The project id that study belongs to

User Table (USER)- Contains Information on Crash Magic users.

- Name -(50) Character field- Name of user
- Description -(255) Character field - Description of the user
- DateCreated -Date field- Date user record was created
- DateModified -Date field- The last date the user record was modified
- DateAccessed - Date field- The last date the user record was accessed for archive purposes
- ParentID -Number field- Contains the parent id of the current user record
- ViewState - Number field- Stores the view state code for the current user
- CurTab -(50) Character field- The current tab the user record is displaying
- ObjType -(50) Character field- The type of object the program will store the record in
- EstXMLSize -Number field- Size of the XML data stored in the XMLData field
- XMLData -CLOB field- Specific XML data for the user record
- UserGroupID -Number field- The group that the user record belongs to
- UserID -Number field- Number assigned to each user in the system
- ULogin -(50) Character field- The login for the current user
- UPassword -(50) Character field- The encrypted password for user
- CanAdmin -Number field- Administrator access flag for the user
- CurAdminUserGroupID - Number field- The current admin user group id
- CurAdminUserID - Number field- The current user ID of the admin
- CurAdminPSRAAttrID - Number field- The current PSRAAttr ID of the admin
- CurUserGroupID - Number field- The current user ID of the login
- CurUserID -Number field- The current user of the login
- CurProjectID -Number field- The current project ID of the login
- CurStudyID -Number field- The current study of the login
- CurReportID -Number field- The current report of the login

Document Data Table (DOCDATA)- Stores supplemental documents added by users.

- Name -(50) Character field- Name of the report
- Description -(255) Character field - Description of the report
- DocID -Number field- The id to identify the document stored
- Document -BLOB field- Binary object
- DateCreated -Date field- Date document record was created
- DateModified -Date field- The last date the report record was modified
- BlobFormat -(255) Character field - The format the BLOB is stored in
- DateAccessed -Date field- Last date the record was accessed for archive purposes

Document Reference Table (DOCREF)- Stores references to documents stored in Crash Magic.

- Name -(50) Character field- Name of document reference
- Description -(255) Character field - Description of document reference
- DateCreated -Date field- Date document reference record was created
- DateModified -Date field- The last date the document reference was modified
- ParentID -Number field- Contains the parent of record of the document reference
- ViewState -Number field- Stores the view state code for the current document reference
- CurTab -(50) Character field- The current tab the document reference is displaying
- ObjType -(50) Character field- The type of object the program will store the record in
- DocRefID -Number field- The id to identify the document reference stored
- DocID -Number field- The document id the reference is pointing to
- UserGroupID -Number field- The group the reference belongs to
- UserID -Number field- The user the reference belongs to
- ReportID -Number field- The report the reference belongs to
- StudyID -Number field- The study the reference belongs to
- ProjectID -Number field- The project the reference belongs to
- PSRAtrID -Number field- The PSRAtr the reference belongs to
- FromToRuleID -Number field- From to rule used by the reference
- EstProcessTime -Number field- Estimated time to process the reference
- EstXMLSize -Number field- The estimated size of the XML data stored in the record
- DateAccessed -Date field- Last date the record was accessed for archive purposes

From To Rule Table (FROMTORULE)- Stores rules for changing displayed data.
(Mainly used for street name changes)

- FromToRuleID -Number field- The id of the From To Rule
- DateCreated -Date field- Date the From To Rule was created
- DateModified -Date field- Date the From To Rule was last modified
- FromRule -(255) Character field- The from criteria for the rule
- FromUnique1 -(255) Character field- A unique criteria to apply the From rule to
- FromUnique2 -(255) Character field- A unique criteria to apply the From rule to
- ToRule -(255) Character field- The to criteria for the rule
- ToUnique1 -(255) Character field- A unique criteria to apply the To rule to
- ToUnique2 -(255) Character field- A unique criteria to apply the To rule to
- DateAccessed -Date field- Last date the record was accessed for archive purposes

Replacement Reference Table (REPLACEREF)- Stores information referencing replacement categories to From To Rules

- ReplaceRefID -Number field- The number that identifies the replacement reference
- FromToRuleID1 -Number field- The From To Rule referenced
- FromToRuleID2 - Number field - The From To Rule referenced (used for intersection aliases)
- ReplaceCatID -Number field- The replacement category referenced
- DateCreated -Date field- Date replacement reference was created
- DateModified -Date field- Date the replacement reference was last modified
- DateAccessed -Date field- Last date the record was accessed for archive purposes
- LocType -Number field- The program constant for type of alias (Intersection, street name, node, etc)
- Purpose -Number field- The reason the alias was entered (Same location, misspelling, etc)

Crash Magic indexes

Crash Magic system tables use indexes to speed up searches. These indexes can be created with the system tables or created manually along with the table creation script. Each index must start with a prefix of PK for primary key or an IX for index. The index name will then contain the the table prefix name entered in the configuration editor followed by the table name the index is created for. The last portion of the index name is the column name of the index. DB2 will use an abbreviated table name and column name in the index name.

The following example is a list of index names created by Crash Magic with a table prefix of cm:

User table

- PKcmUserUserID
- IXcmUserName

- IXcmUserUserGroupID
- IXcmUserULogin

UGroup table

- PKcmUGroupUserGroupID
- IXcmUGroupName
- IXcmUGroupSharedUserID

PSRattr table

- PKcmPSRattrPSRAttrID
- IXcmPSRattrName
- IXcmPSRattrUserGroupID
- IXcmPSRattrUserID

Project table

- PKcmProjectProjectID
- IXcmProjectName
- IXcmProjectUserGroupID
- IXcmProjectUserID

Study table

- PKcmStudyStudyID
- IXcmStudyName
- IXcmStudyUserGroupID
- IXcmStudyUserID
- IXcmStudyProjectID

Report table

- IXcmReportName
- IXcmReportUserGroupID
- IXcmReportUserID
- IXcmReportStudyID
- IXcmReportProjectID

CID table

- PKcmCIDCIDID
- IXcmCIDName
- IXcmCIDUserGroupID
- IXcmCIDUserID
- IXcmCIDStudyID
- IXcmCIDProjectID

Help system

HelpSystemOverview

The Crash Magic help system is designed to be context-sensitive. The program has, at all times, a list of keywords based on the current user actions to that point. When help is activated, that list is scanned and appropriate help topics are matched up with the keywords. Any matches are presented in the help box for selection.

Customizing the help system

The core help system is based on the Crash Magic documentation. However an end user administrator may add additional content to any help keyword by editing the helpreference attribute and adding a help topic for that keyword.

The structure of the helpreference attribute is an xml document like this:

```
<pdroot>
  <HelpReferenceTable>
    <General>
      <BaseURL>/files/Help/HTML</BaseURL>
    </General>
    <HelpItems>
      <!-- ***** Analysis side help keywords ***** -->
      <HelpItem Keyword="cmMainForm">
        <Title>Crash Magic Online</Title>
        <Description>Basic information about Crash Magic</Description>
        <URL>IntroductionOverview.htm</URL>
        <OtherKeywords>
          <Keyword>CurrentWebManual</Keyword>
          <Keyword>PDFManual</Keyword>
          <Keyword>Tutorial</Keyword>
          <Keyword>PSROverview</Keyword>
        </OtherKeywords>
      </HelpItem>
      <HelpItem Keyword="cmDefaultPanel">
        <Title>Home Page</Title>
        <Description>Home page</Description>
        <URL>Homepagepanel.htm</URL>
      </HelpItem>
      ...
      <HelpItem Keyword="cmAdminPSRattrPanel">
        <Title>Attributes information</Title>
        <Description>Generic attributes panel information</Description>
        <URL>Attributes.htm</URL>
      </HelpItem>
      <HelpItem Keyword="cmAdminPSRattrPanel">
        <Title>Attributes information</Title>
        <Description>Generic attributes panel information</Description>
        <URL>Attributes.htm</URL>
      </HelpItem>
      <!-- ***** User-defined help keywords ***** -->
      <HelpItem Keyword="UserDefined">
        <Title>User-specific help topic</Title>
        <Description>Base level user defined help link</Description>
        <URL>help_system_description.htm</URL>
      </HelpItem>
```

```
</HelpItems>
</HelpReferenceTable>
<Settings/>
</pdroot>
```

The BaseURL tag defines where to look for a URL if the URL isn't already fully qualified. This makes it possible to direct an installation to point to the Pd' Programming server and have up to date help all the time, or to host the help files on any other server. The default is /files/Help/HTML which points to the help system installed with the application.

Each HelpItem is referenced by a "Keyword" that is specified by the application. That is, the application adds keywords to the help list as the program runs, and when help is called for, this list is searched for a HelpItem that specifies that Keyword. It then displays the "Title" in the help box and links it to the "URL" specified.

For every case where a Keyword is added to the help list, that keyword is also added with a "_Custom" as well. This makes it possible to create end-user defined help and have it appear from within the same help system as the built-in help. In this case, the URL will probably be fully qualified and point to a user-maintained html or other document.

Rather than edit the "Default" helpreference in the .master/.shared attribute list, the preferred method is to create another "Default" helpreference in the appropriate User Group / .shared attribute list. (copy/paste is an easy way to do this) Then simply add _Custom to any item you wish to add additional help to. When the help system looks for a help keyword, it first looks in the User's attributes, then the group's .shared attributes and finally in the .master/.shared attributes. It displays the first help it finds.

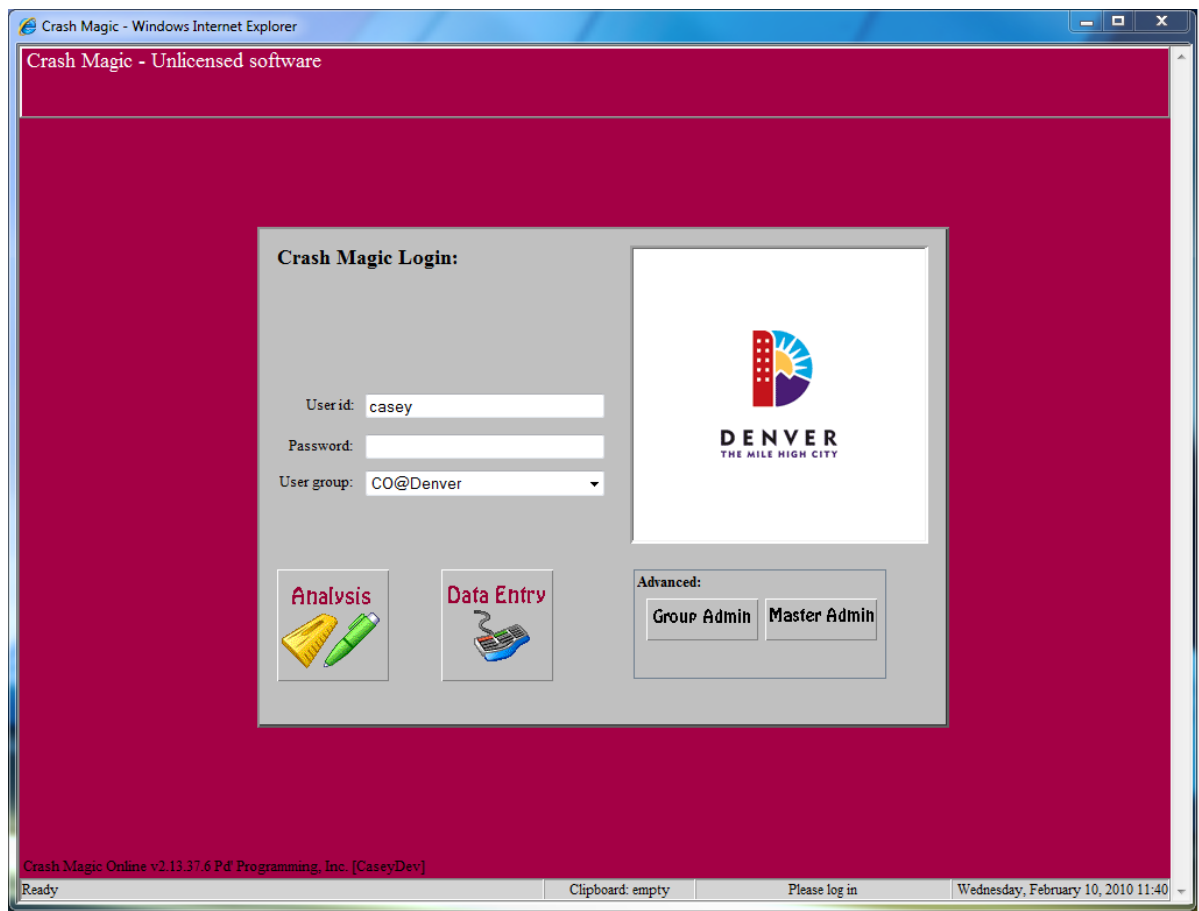
Login Methods

A basic requirement of the program is that each user must have their own "account". The account places the user within a group, which provides configuration and setting information. The account is also used to provide a repository to load and store templates, settings, reports, etc. A user may only be logged in one at a time. If a user attempts to log in to the system while an existing session is still active for them, they will be denied access unless they choose to terminate the existing session.

In order to identify a user and match them up with their account, a login is required. Since different agencies have different security and system configuration needs, Crash Magic supports several login methods.

Interactive login page

The most straightforward login method is to browse to the home page of the application where the user is presented with prompts for Login name, password and user group.



Upon selecting either "Analysis" or "Data Entry" or one of the administration buttons, the system simply searches for a login with the appropriate name and password within the specified user group. If the login is found, it is made current and the user is sent to the desired page. Otherwise, the user is informed of the error and they may try again.

Magic Auto login

The MAGICAUTO url command supports passing login information in order to bypass the login screen. For more information on this, please see the [Automatic Login](#) ³⁵⁰ section in Automation and interoperability.

Database passthrough

In some situations each user is provided a database login. Specifically, each user has an account in the Oracle, MS or other SQL server. While each user must have a Crash Magic account, the login validation can be performed by your SQL database. This is accomplished by creating a simply query which returns few, if any rows.

To accomplish this, create a query in the group's .shared account called "DBVerify". The connection for that query will generally be the same connection as is used by all the user queries. (usually "Default") The DBVerify query should issue a very simple query

against a small table in the database that valid users have access to, and others will not.

An example query might be:

```
SELECT LookupColumn  
FROM DirectionLookups  
WHERE 1=0
```

This query will never return any rows. However, it will raise an error if the user doesn't have database permission to issue the query. So, how does the system know the user?

The secret is in the connection object. The connection object used for all user queries should specify %USER.LOGIN% for the login field. The password field may be left blank. (this can be easily set by clicking on the "Set for DBVerify login" link on that screen.

The result is that upon login, the system records the user's login name and password and user group. The login name and password are used to create a database connection, and the DBVerify query is issued. If the query succeeds, the user is logged in to the Crash Magic account they specified by user group and login name. In addition, the password they specified is maintained in memory to re-create the connection as needed. The password is never stored on disk or database, just in memory and just for the duration of the user's session.

In this way, each user is logged into their own account on the SQL server, and if validated, they are mapped to the correct Crash Magic account. When passwords change on the SQL server, nothing needs to be done in Crash Magic, as the passwords are simply passed-through to the SQL server for validation.

Active directory

Logging in via Active Directory (a MS Windows domain-based technology) is available when running Crash Magic as an ISAPI inside MS IIS. It also requires that the user has already presented login information to their browser prior to accessing the Crash Magic login page. If this has been done, then login information is passed from IIS directly to Crash Magic.

The login information passed from IIS includes the domain and the user name. The password is not sent to Crash Magic. What Crash Magic does is look for the user name in all the user groups that have an "AuthDomains" list that includes the specified domain name. If a user is found in one of these groups, the user is logged into Crash Magic. This method of validation works because the login and domain name are not sent unless the user has logged into a valid Windows domain.

The .master user group should never be set up with Active Directory login. The default behavior for Crash Magic is to login the user to the analysis section of the program. This will prevent a user from logging into the .master user group.

Note: Once AuthDomains is populated with a valid domain, users will not see the login screen when they access the main Crash Magic URL. A new URL parameter has been introduced when "AUTOLOGIN" is set to false (i.e. <http://somedomain.com/cm/?AutoLogin=false>) the user will be presented the login page and no automatic login action will be taken. Also, when logging out of the program, this parameter is sent automatically - otherwise it would be impossible to log out. It is suggested that system administrators create a link such as this to access the Crash Magic system without being logged in automatically.

Set Crash Magic to use Active directory:

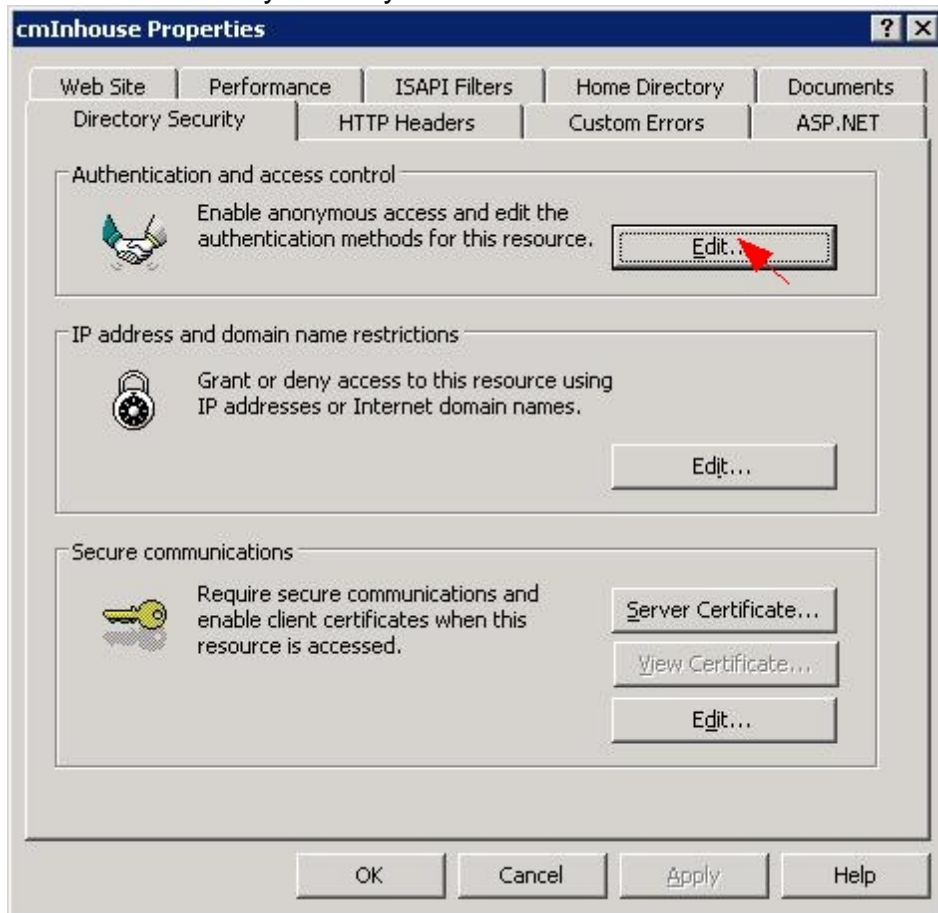
1. Log into Crash Magic as the Group Admin
2. Click on your the user group in the project tree that you have logged into (<state or province>@<municipality>)
3. Ensure the green Settings tab is open
4. Enter the name of the domain that Crash Magic is running under in in the Auth domain field

The following steps assume that you have IIS installed and running.

Steps setting Active Directory Logins on IIS 6:

1. On the server hosting Crash Magic open the Internet Information Services Manager
2. Right click on the web site or virtual directory Crash Magic is deployed under
3. Select Properties

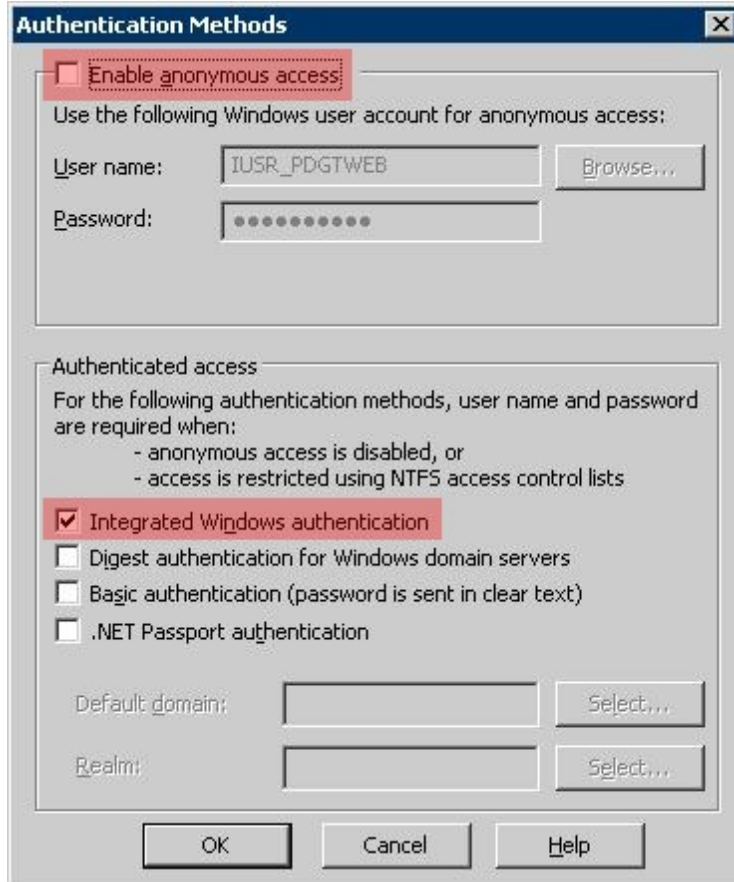
4. Select the Directory Security tab



5. Click on the Edit button for Authentication and access control

6. Uncheck the Enable anonymous access

7. Check Integrated Windows authentication



8. Click the OK button to save the work

Steps for setting Active Directory Logins on IIS 7:

1. On the Crash Magic server ensure that the Web Server/Security/Windows Authentication role has been installed(Consult the Windows documentation to install this role if needed)
2. Open the IIS Manager
3. Click on the Crash Magic application home(cm is the default home)
4. Click on Authentication
5. Right click on Anonymous Authentication and select Disabled
6. Right click on Windows Authentication and select Enabled

Enable Crash Magic users to access the Sys directory for IIS 6 and 7:

1. On the server hosting Crash Magic create a cmUserGr group in windows groups
2. Grant full control to the cmUserGr group to the Sys folder and all of the subdirectories
3. Make the cmUser a member of the cmUserGr group
4. Add any windows users groups that will access Crash Magic to the cmUserGr group

Integrated Windows Authentication on client browsers:

Networks that are using Kerberos authentication must ensure that "Enable Integrated Windows Authentication" is enabled. Networks that use NTLM should disable this setting. This setting is available in the advanced tab of the internet options dialog box in IE.

1. Open IE
2. Click on the Gear symbol near the upper left corner of the window
3. Select Internet Options
4. Click on the Advanced tab
5. Scroll down to the Enable Integrated Windows Authentication
6. Check the box according to your network authentication(Uncheck for NTLM Checked for Kerberos)
7. Click OK
8. Close and reopen the browser

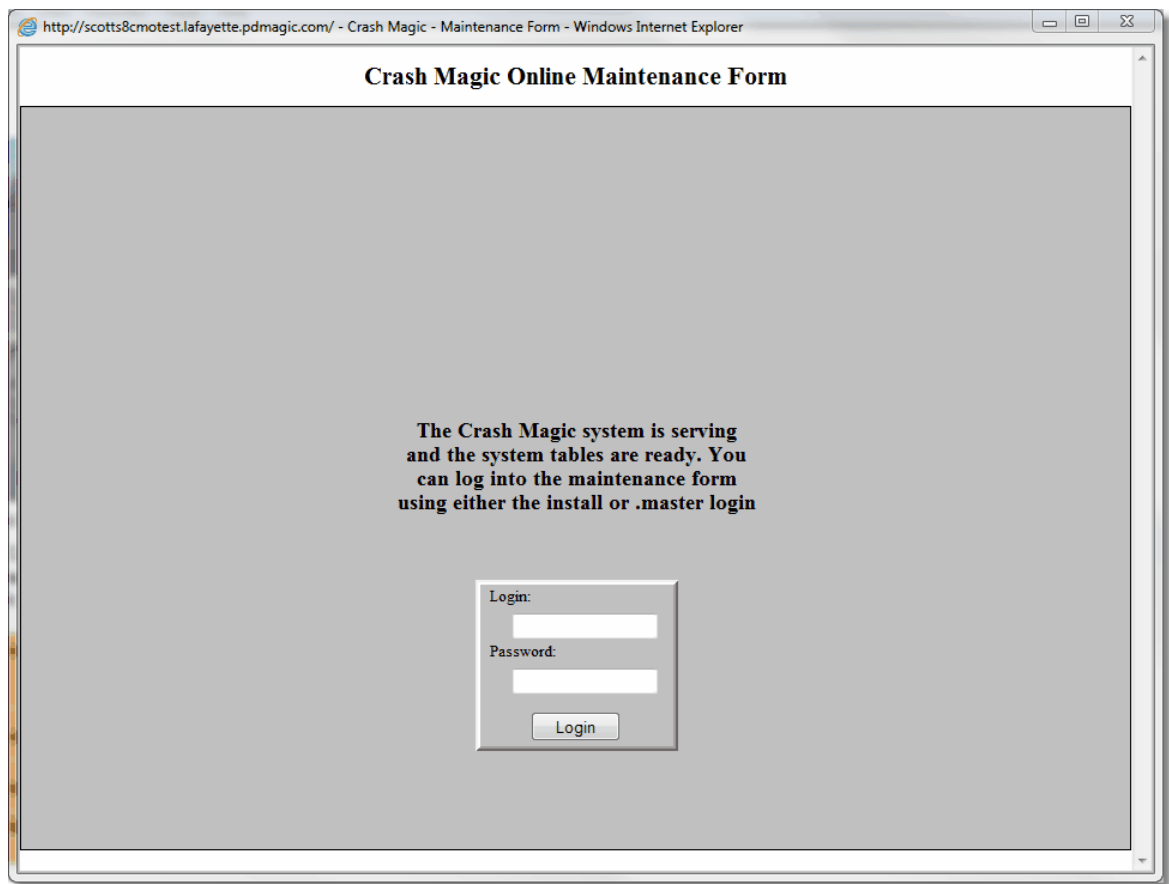
Set the appropriate User Authentication for the Crash Magic site when required:

Incorrect IE Security settings can block a user from being able to log into Crash Magic. A Logon value of Anonymous will ensure the user is not able to log into Crash Magic. The browser will also prevent a login to Crash Magic if Automatic logon only in Intranet Zone and IE does not detect that Crash Magic is within the Intranet.

1. Open IE
2. Enter the Crash Magic url, and go to the site
3. Click on the Gear symbol near the upper left corner of the window
4. Select Internet Options
5. Click on the Security tab
6. Click on the Custom level
7. Scroll to the User Authentication/Logon section
8. Set the appropriate Logon level(This radio button should be set to Automatic logon with current user name and password will or Prompt for user name and password)

Maintenance Form

The Crash Magic Maintenance form allows control of the Crash Magic application. This is the screen seen when a group admin first opens the Crash Magic web page. The initial login and password are set from the Crash Magic - Post install utility. Once the group admin has completed setting up the Crash Magic server the Master Admin login can also be used to log into the Maintenance Form.



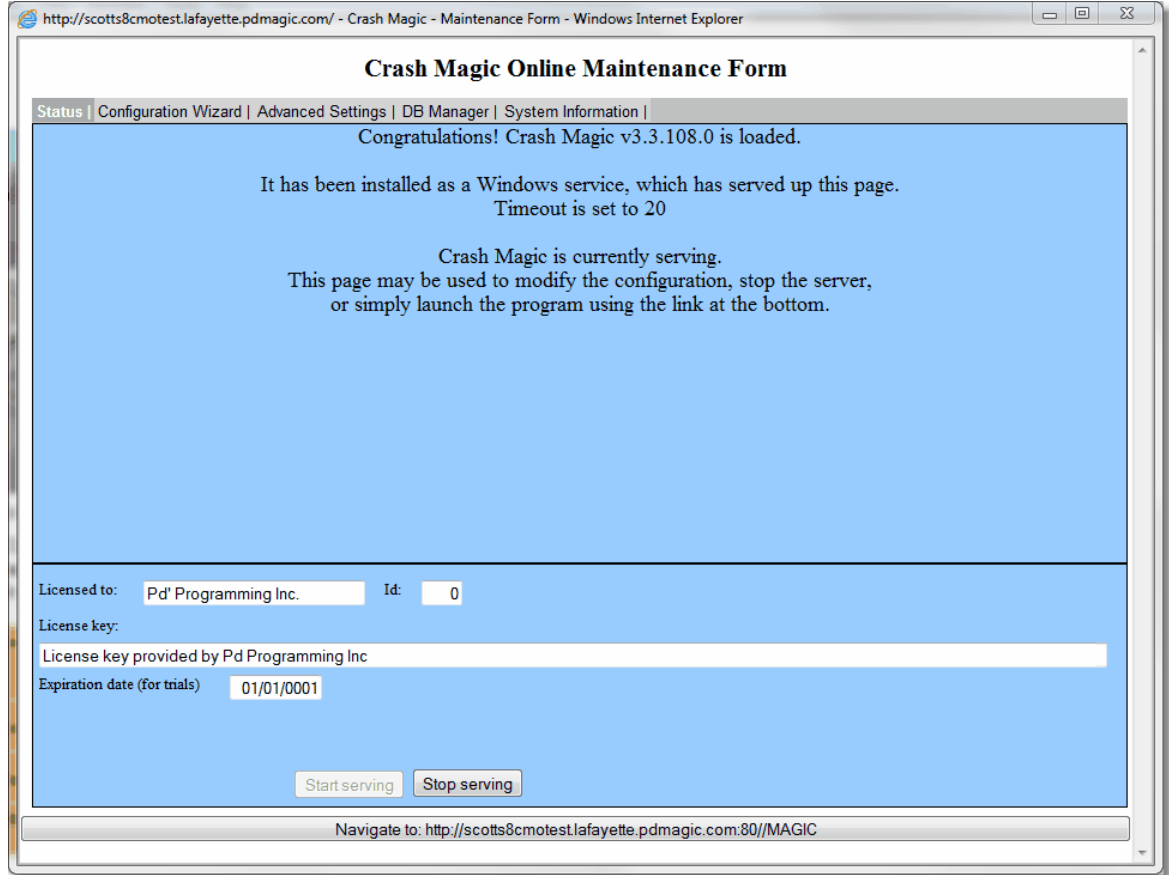
Five tabs are used to display information on Crash Magic once logged into the Maintenance form

- Status
- Configuration Wizard
- Advanced Settings
- DB Manager
- System Information

Status Tab

The Status tab of the Maintenance form provides information on the Crash Magic server. It provides messages, registration, and allows the Crash Magic server to be started and

stopped



- **Information section** - This section will display the version information, user timeout setting and the current status of the application. This main window will also contain any messages relevant to installations and upgrades.
- **Licensed to** - Shows the name of the organization the current Crash Magic version is licensed to. The Licensed to name must match the name, ID and Key provided by Pd' Programming with the purchase of the software.
- **ID** - This is the client ID provided by Pd' Programming.
- **License Key** - This is the license key provided by Pd' Programming to run the software.
- **Expiration date** - This is the expiration date of trial software.
- **Start serving** - Starts the Crash Magic server. This button will be grayed out when the server is running.
- **Stop server** - Stops the Crash Magic server. This button will be grayed out when the server is not serving. This button should be used with care as it will kill sessions for users currently logged into Crash Magic. If the server has stopped users will only see the Maintenance Form login page.
- **Navigate to** - This button will open the login page when the Crash Magic server is running.

Configuration Wizard Tab

The Configuration Wizard tab is used to set up the Crash Magic system tables. Crash Magic uses a group of tables to store information. These system tables store information on mapping a collision database to the Crash Magic program. This wizard is used to define a connection, and create the system tables.

The screenshot shows a web browser window titled "Crash Magic - Maintenance Form - Windows Internet Explorer". The address bar shows "http://scotts8cmotest.lafayette.pdmagic.com/". The page title is "Crash Magic Online Maintenance Form". The navigation bar includes "Status | Configuration Wizard | Advanced Settings | DB Manager | System Information |". The main form has several sections:

- Application Directory:** C:\Program Files (x86)\PdMagic\CrashMagicOnline\Bin
- System files Directory:** C:\ProgramData\PdMagic\CrashMagicOnline\Sys
- Samples:** MS Provider for SQL Server (dropdown menu)
- Use sample as connection string:** (button)
- SYS Connection string:** (text box)
- SQL Server type:** MSSQL2008 (dropdown menu)
- Login:** (text box)
- Password:** (password box with dots)
- Role:** (text box)
- Password:** (password box with dots)
- Schema owner and prefix (i.e. dbo.Cm):** (text box)
- Create tables:** (button)
- Valid admin account:** (text box)
- Master login:** (text box)
- Master password:** (password box with dots)
- Valid master config:** (text box)
- Valid custom config:** (text box)

The status bar at the bottom says "Navigate to: http://scotts8cmotest.lafayette.pdmagic.com:80//MAGIC".

- **Application Directory** - Shows the install location of the Crash Magic binary files.
- **System Files Directory** - Shows the location of the files used by Crash Magic.
- **Sample** - This selection list allows the selection of a sample connection string to the Crash Magic system tables.
- **Use sample as connection string** - This button populates the SYS connection string box with a connection string based on the Samples selection. The SYS connection string box must be blank for the box to be populated.
- **SQL Server type** - Specifies the type of database that Crash Magic will be connecting to for the system tables.
- **Login** - The login name that Crash Magic will use to access the system tables.
- **Password** - The password used by the login for the Crash Magic system tables.
- **Role** - Role that will be used by the above login to access the Crash Magic system tables. If no role will be used this box should be left blank
- **Password** - This is the password for the role that will be used by the role. As with the role this field should be left blank if no role will be used.

- **Schema owner and prefix** - This is the schema under which the Crash Magic system tables will reside and the prefix of system tables name(The recommended table prefix name is cm Example dbo.cm where dbo is the name of the schema that the tables will reside, and the Crash Magic system tables will all start with cm in their names). This field is case sensitive. The schema and table prefix must match the case of the schema and table names.
- **Create tables** - This button will open the window the generates the script to create the Crash Magic system tables.
- **Master login** - This field creates the Master Admin for Crash Magic. This login should be retained for Support issues, and should only be used with the guidance of Pd' Programming.
- **Master password** - Password that will be used for the Master login. This should be retained along with the Master login.
- **Valid master config (Update .Master/.Shared configuration)** - This button will populate the Crash Magic system tables with information that is standard
- **Valid custom config (Browse)** - This button allows the administrator to select the configuration provided by Pd' Programming. This configuration file will tell Crash Magic how to display the collision data that will be used.
- **Valid custom config (Import configuration)** - This button imports the configuration selected by the previous browse button into Crash Magic.

The Create table window allows the database administrator to create the tables and indexes required by Crash Magic.

Crash Magic Online Maintenance Form

Status | Configuration Wizard | Advanced Settings | DB Manager | System Information |

Create tables

Display SQL and then Execute SQL. The "access" and "namespaces" parameters are optional and will be applied to the new system tables if specified. The "login" and "role" values are used to create the new tables and must have the appropriate permissions.

Grant access to: Login:

Tables namespace (optional): Password:

Index namespace (optional): Role:

Navigate to: <http://scotts8cmotest.lafayette.pdmagic.com:80/MAGIC>

- **Grant access to** - Specify the database user that Crash Magic will use to access the system tables. This user will be granted Select, Insert, Update and Delete to the system table records.
- **Tables namespace** - This allows oracle users to specify a name space for the tables. This field should be left blank if no namespace is required or specified.
- **Index namespace** - This allows oracle users to specify a name space for the indexes. This field should be left blank if no namespace is required or specified.
- **Display SQL** - This button will generate the SQL statements to create the Crash Magic system tables and indexes. These statements will be shown in the text box below. Database administrators can choose to copy the SQL and run the statements from their own database interface. Crash Magic does not allow changes to the table, and index structures, but storage clause information can be added.
- **Login** - This is the database login of the user that will create the tables and indexes for Crash Magic. This user must be able to create tables in the schema specified from the previous window.
- **Password** - This is the password for the login above
- **Role** - This allows the user to specify a role required by the login above to create the Crash Magic system table and indexes. This should be left blank if no role is required.
- **Password** - This is the role password. This should be left blank if no role or password is required.

- **Execute SQL** - This will execute the SQL show in the bottom text box. Connection information from the configuration wizard tab will be used to execute the SQL displayed.
- **Done** - This button will close the Create tables window.

Advanced Settings Tab

The Advanced Settings tab allows control over server settings that are not generally changed by the client.

The screenshot shows the 'Crash Magic Online Maintenance Form' in a web browser window. The browser's address bar shows the URL: `http://scotts8cmotest.lafayette.pdmagic.com/ - Crash Magic - Maintenance Form - Windows Internet Explorer`. The form has a title bar and a navigation menu with tabs: 'Status', 'Configuration Wizard', 'Advanced Settings' (selected), 'DB Manager', and 'System Information'. Below the menu, there is a 'Session timeout (in minutes):' field with the value '20'. The form is divided into several sections: 'Settings ignored in IIS / ISAPI:' (with sub-sections for 'Limited server IP:' and 'Server port: 80' and a 'Use compression' checkbox), 'Settings for ISAPI only:' (with 'Thread pool min:' and 'Thread pool max:' both set to 3), 'Debugging:' (with checkboxes for 'SQL SYS logging', 'SQL USER logging', and 'Use JS for popups', and dropdowns for 'COM initialization: Default' and 'Log level: Important'), 'System database connection:' (with 'Pool min:' set to 3, 'Pool max:' set to 20, and checkboxes for 'Force uppercase schema names' and 'Ignore incorrect schema info for:'), and 'Error reporting:' (with a 'Post' dropdown and a 'Test' button). At the bottom, there is a text box with the URL `http://www.pdmagic.com/Exceptions/index.cfm` and a 'Navigate to:' label. The browser's status bar at the bottom shows the URL `http://scotts8cmotest.lafayette.pdmagic.com:80/MAGIC`.

Session timeout (in minutes) - Specifies the length of time a user session can remain inactive until Crash Magic makes the user log back in.

The "Settings ignored in IIS/ISAPI" section contains items that are only for use when Crash Magic is run as a stand alone service. These items are not relevant when running Crash Magic under IIS.

Limited server IP - Limits the Crash Magic web service response to a single IP.

- **Server port** - The port that Crash Magic will serve web pages on. This defaults to the standard port of 80
- **Use compression** - Determines whether information should be compressed when sent to the browser.

The Debugging section should only be changed with the advice of Pd' Programming support as these items can affect Crash Magic performance.

- **SQL SYS logging** - This check box determines whether logging of SQL statements to the Crash Magic system tables should occur. The default of this option is uncheck so that no logging will occur.
- **SQL User** - This check box is used to log SQL statements sent to the collision database. The default option for this box is unchecked so that no logging will occur.
- **Use JS for popups** - This check box tell Crash Magic to use java script for popups. This means that users will receive java popup windows when interacting with Crash Magic.
- **COM initialization** - Sets how COM is initialized in Crash Magic. The default setting for this menu is Default.
- **Log level** - This defines what information will be logged to the Crash Magic system logs. The default level is important.

Crash Magic can send information on errors that users encounter to Pd' Programming. The error reporting section is for adjusting those settings. The first drop down menu specifies how the error messages are to be sent to Pd Programming. This defaults to Post. The text box below specifies the location a message will be sent to when the Post report is selected.

- **Test** - Will send a test error message to Pd Programming.

The Settings for ISAPI only will only be used when Crash Magic is run under IIS.

- **Thread pool min** - Sets the minimum number of thread pools that will be used by Crash Magic when running under IIS. The default is 3.
- **Thread pool max** - Sets the maximum number of thread pools that will be used by Crash Magic when running under IIS. The default is 3.

The System database connection section allows for control of advanced connection options to the Crash magic system database.

- **Pool min** - The minimum number of connections that will be established to the system database.
- **Pool max** - The maximum number of connections that will be used by Crash Magic. While Crash Magic does reuse the connections to the system tables, multiple user environments may require a greater number of connections to avoid user wait times on connections.
- **Force uppercase schema names** - This check box forces Crash Magic to use upper case names when querying the Crash Magic system tables. This box is checked by default.
- **Ignore incorrect schema info for** - This text box allows the user to specify the columns and indexes in the Crash Magic system table that should not be validated. This text box will take one line per item. Items should be specified in a schema.table.column or schema.table.index notation. *Note: Pd' Programming has found that in some cases DB2 will report CLOB data types differently in ADO than what is displayed to the average user. In these cases the client can specify columns and indexes to skip validation.*

DB Manager Tab

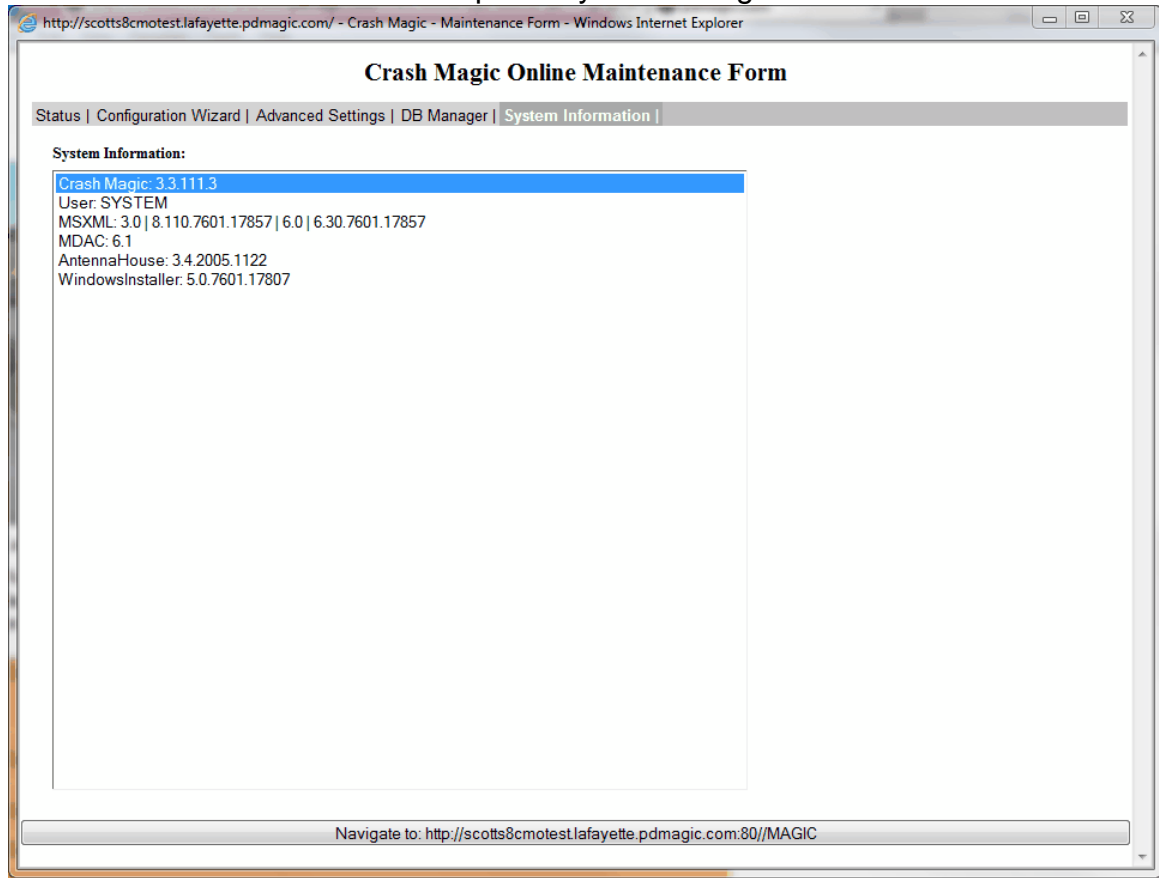
This tab is used to manage the Crash Magic system tables data. This form should only be used with the advice of Pd' Programming support.

The screenshot shows a web browser window titled "Crash Magic - Maintenance Form - Windows Internet Explorer". The address bar shows the URL "http://scotts8cmotest.lafayette.pdmagic.com/". The page title is "Crash Magic Online Maintenance Form". The navigation bar includes links: "Status | Configuration Wizard | Advanced Settings | DB Manager | System Information". The main content area has a warning message: "Warning: Controls on this page are used to destroy system tables data and the actions can not be reversed. Please check the box below to confirm that you understand and would like to proceed with deleting data." Below the warning is a checkbox labeled "Check this box before selecting an option below." To the right of the checkbox is a button labeled "Export system table data". Below the checkbox are four buttons: "Drop all system tables", "Delete master group", "Delete all system data", and "Delete all .master users". At the bottom of the page, there is a navigation bar with the text "Navigate to: http://scotts8cmotest.lafayette.pdmagic.com:80//MAGIC".

- **Check this box before selecting and option below** - This box must be checked before clicking any of the buttons below as they can delete data from the Crash Magic system tables.
- **Drop all system tables** - This will drop the Crash Magic system tables deleting any data stored within them.
- **Delete the master group** - This will drop the .master group in the Crash Magic system tables. The .master group contains default information for Crash Magic, and the program is unable to run without this information.
- **Delete all system data** - This button will delete all of the records from the Crash Magic system tables. The table structure will remain, but all of the records will be deleted.
- **Delete all .master users** - This will delete all of the .master users from the tables. This will allow the .mater admin user to be reset in the Configuration wizard tab.
- **Export system table data** - This button will export all of the Crash Magic system table records to an xml file.

System Information Tab

The system information tab displays version of Crash Magic that is currently running, and version information of software required by Crash Magic.



Multiple program instances

It is sometimes desirable to install multiple program instances on a single computer. Crash Magic supports this when installed as a service or when installed as an [ISAPI into IIS](#)¹⁴⁹. This is an advanced topic. Installing multiple instances of Crash Magic should only be undertaken after consulting your DBA as well as your web server administrator.

This feature makes possible access of the program on a single computer with different URL's. For example:

Under IIS:

<http://servername/cmbeta>

<http://servername/cmproduction>

<http://servername:81/cmbeta>

<http://servername:80/cm>

This feature can be useful when hosting Crash Magic for multiple clients. Crash Magic is capable of hosting multiple configurations and user groups with a single instance.

However, using multiple instances can isolate each client's process and avoid having malfunctions on one system affect another.

This feature is also very useful when testing an installation. (i.e. moving from testing to production) By using two different instances, you can host two different versions of the program on the same system.

Each Crash Magic installed instance resides in it's own pair of folders. The base folders are c:\program files (x86)\pdmagic and c:\programdata\pdmagic. Each instance get's its own sub-folder named cmo<instancename>. The program determines where it is being launched from and finds it's paired ProgramData folder.

Multiple IIS instances:

The steps for installing Crash Magic as an ISAPI in IIS are described in the Getting Started chapter. Repeat the [steps for installing in IIS](#)¹⁴⁹ for each instance.

Notes:

- Since IIS manages IP address and ports, these values are ignored when installed as an ISAPI.
- It is generally not necessary to create a new user and/or application pool for each instance. These are often shared.
- A second application pool can be useful if one instance is a non-robust beta instance and the main instance needs to stay running in all cases.
- Since each instance is created by copying the application dll to its own virtual folder, each dll must be registered as its own service extension. (as described in the steps for installing in IIS 6)
- Each instance will have it's own CrashMagic.ini in the virtual folder. Typically this ini only contains the name of the configuration within the CrashMagic.xml configuration file. In some cases it may be desirable to use a second configuration file. (i.e. if two versions of the program use incompatible configuration files) In this case, the CrashMagic.ini file may also include a CfgFileName setting. The CrashMagic.ini file would then have the following format:

```
[General]
CfgName=Default
CfgFileName=c:\program files\pdmagic\CrashMagicBeta\CrashMagic.xml
```

- Each instance must use different system tables and/or a different database. It is not safe for multiple instances to have the same SYS tables.
- Each instance must have a different log files directory
- Each instance must have a different cache directory

PSRattrs

The term "PSRattr" refers to Project Study and Report Attributes. They are also referred to as "resources" or "attributes". It is the information used by the program to be able to interpret and analyze the crash data of a specific client or data source. PSRattrs may be shared among users, groups and even the entire installation. Crash Magic includes as part of its installation a User Group called ".Master" that includes PSRattrs for all configurations. These are stored in the ".config" user and the ".shared" user.

- .config is a user account that is created and maintained by Pd' Programming. Our clients/end users have read-only access to these PSRattrs. This is done to assure that the configurations we prepare continue to function correctly regardless of user actions. Only the required PSRattrs are stored in the .config. Additional or optional PSRattrs are stored in the .shared container.
- .shared is a user account contains PSRattrs that are created by Pd' Programming or the client and maintained by either. These PSRattrs are not critical to the configuration, though often contain reports or parts of reports desired by the client.
- .tools is a user account that is created by Pd' Programming for our own use. The PSRattrs in this account are not visible unless a user account specifies "use .tools". Resources in this account are often those created for a single use or for testing or similar.

This section enumerates the available Crash Magic PSRattrs. The title of each section includes the common or "friendly" name generally displayed to the user, as well as the internal name used by remote access and stored in the database. For example "Normalizer - Streets (normalize-streets)".

PSRattr inheritance

Crash Magic utilizes a powerful inheritance mechanism for selecting resources when running the program. When a user requests a resource such as a filter, the program looks for a filter with that name in a number of places, in a specific order:

1. The user's own account is the first place it looks. Resources in the user's account are only visible to that user.
2. *The .tools account for that user's group. (Only if that user has the "use .tools" permission. This is generally reserved for Pd' Programming configuration developers)*
3. The .shared account for that user's group. This is the place where clients may place shared resources for all of their users.
4. The .config account for that user's group. This is where the configuration resources created by Pd' Programming are stored.
5. The .tools, .shared and .config users for any user groups listed in the "Options - Inheritance sequence" resource for that user's group. This is a common tool for shared configuration such as CA SWITRS, AZ ACIS, CO DR3447, IA SAVER, as well as configurations for other states with common configurations.
6. The .tools, .shared and .config users for the .Master group. These are resources maintained only by Pd' Programming and shared with all client configurations.

The program uses the resource with the same name and type as requested. (e.g. "Clear" Filter)

There are a few handy extras that work within the inheritance mechanism:

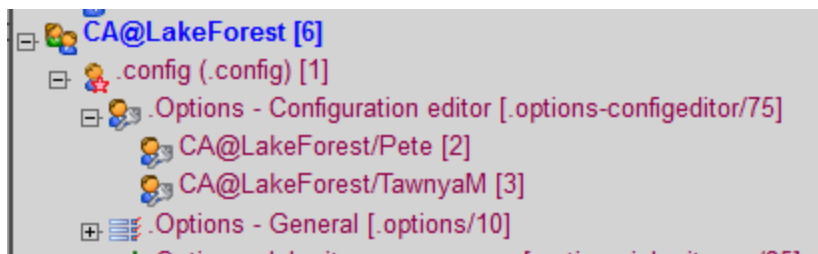
- A. Calculated fields - If an individual calculated field is redefined, the one found first will be used. For example, if a calculated field called "ZipCode" is defined in the .Master/.config as "07901" but is redefined in another user group such as CO@Boulder with the value "80302", any user in the CO@Boulder group will see

ZipCode as 80302. The inheritance of the entire "Calculated fields" resource functions normally. This only applies to fields defined in that record.

- B. Query - A query may specify a "Query - Part" that it will add to. So there may be a Query - Part called "DR3447Crashes" in the _CO@DR3447 configuration, and a Query called "Crashes" in the CO@Boulder configuration that specifies DR3447Crashes as one of its parts. In this case, the DR3447Crashes will be loaded and the Crashes will have its joins and tables added to it to create a final query.
- C. Object map - Like calculated fields, if a rule is created with the same name as an existing rule created in a shared account, the first rule found will be used. This feature can also be used to remove rules by simply declaring an empty rule with the same name as an existing one.

.Options - Configuration editor (.options-configeditor)

The .config containers hold configuration PSRattrs for a user group or shared user group. Those PSRattrs should only be modified by the person(s) responsible for the configuration. In order to protect it, only those logins registered as "configuration editor" may change their content. To register a login as able to edit the contents of a .config, create a .Options - Configuration Editor with the appropriate group and that login as its name.




Tree showing Configuration editor resources

These resources work simply by existing in the .config container. They do not have any relevant properties to set.

Upon login, a user's account is checked against this list to see if they are a configuration editor.

For Pd' Programming - prepared configurations only Pd' accounts should exist here. There are no exceptions. For a client to modify their configuration, they may copy and paste a PSRattr into the .shared container and edit it there. This maintains the stability of the configuration.

.Options - General (.options)

While under the .shared user of the user group, click the .options button  to define options that a Crash Magic user will be able to use.

Name: Enter Default for the name

Description: A description of the .options.

City: If this configuration is being for a city enter the city name. Else leave this section blank.

State abv: Enter the State or province abbreviation of that the collision data that will be used for this configuration. This field should be left blank if the configuration is for data will be from more than one state or province.

Zip: If this data will only be from a single zip code enter the zip code in this field. Leave this field blank if the collision data will be from more than one zip code.

Email- This section allows group administrators to define mail server information for Crash Magic clients.

SMTP URL: The email url.

SMTP URL: Port used by email program.

Folder Path: This text box shows the folder path of the location selected in the "Folder Locations" box. The folder location can be edited by using this text box. The alias will default to "Location" when first created. This text box can be used to change to the correct location. Example:

```
C:\Program Files\PdMagic\CrashMagicOnline\Bin\ProgramResources\Samples\Images
```

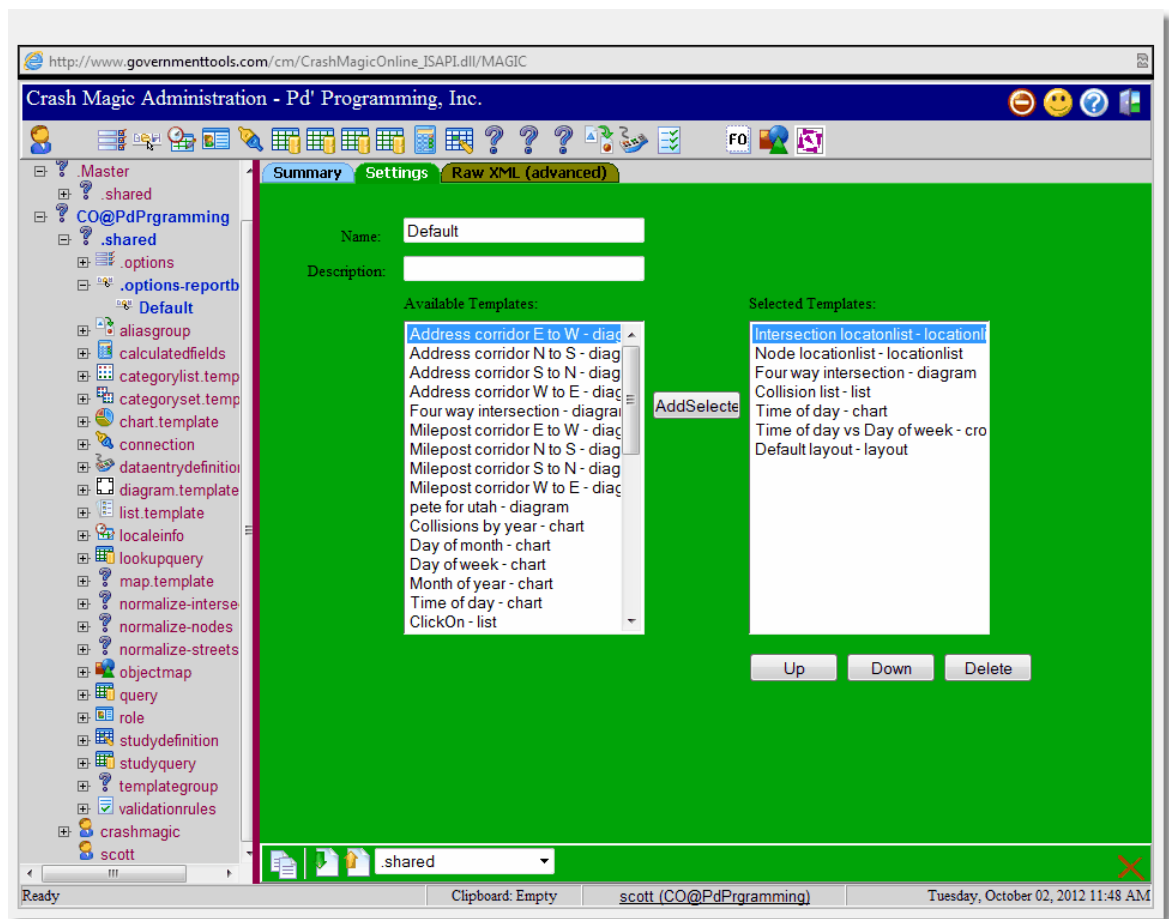
Add button: This button adds the alias entered in the "New Location Name" field to the "Folder Locations" box.

Remove button: This button removes the alias selected in the Folder Locations box.

Test path: This button attempts to locate the folder selected in the selected in the Folder Locations. A message box of "Success: Folder found" will be displayed if the folder can be accessed. "Error: folder does not exist" will be displayed if Crash Magic is unable to access the folder.

.Options - Report buttons (.options-reportbuttons)

This attribute sets the report buttons displayed to the user on the analysis side of Crash Magic.



Available templates - Displays a list that users can select of the current template that can be used as a button to create a report with.

Add Selected - Adds the selected Available template to the Selected Templates list.

Selected Templates - Displays a list of the templates that will be displayed on the user as a report button.

Up - moves the template selected in the Selected templates window up.

Down - moves the template selected in the Selected templates window down.

Delete - Removes the selected template from the Selected Templates list.

Alias group (aliasgroup)

Alias Groups allow the storage of alias street names. While this object is required to store this information the actual editing of street aliases is done from within the [Streets Normalizer](#)²⁶⁶.

Name: This is the name that will be given the alias

Description: This is the description for the alias

Calculated fields (calculatedfields)

Calculated fields allow users to create named expressions. Expressions can be based on the results returned by a query. Some calculated fields have already been created for you in the .Master/.config user, and they can not be altered. Other calculated fields are added to the user group. For a query to use a calculated field it must be checked in the Study Query settings tab. Calculated field groups are loaded in alphabetical order based on the name of the Calculated field group. Calculated fields can use other calculated fields provided they can be used by the query, and that they have already been loaded by virtue of their alphabetically sequenced group name.

- P_Default - These fields are required by Crash Magic. They are used to define required information.
- P2_Default - These fields are required by Crash Magic. They define supplemental location information.
- U_UserSpecific - These fields assist with specific user data calculations. This calculated field group is generally used for simplifying expressions in the object map.
- V_UserDefined - These calculated fields should be created for every configuration, but left empty for the client to populate as desired

The following is a list of the data types that Crash Magic uses and their descriptions.:

Crash Magic data types	Description	Common database data types
Boolean	This field is used for storing True or False values	
Date	Date and time contained in a single field	
Number	Any valid number(whole numbers, integers, decimal and floating point)	
String	Any character string value	
XML	Valid XML	

The P_Default calculated fields are standard fields used by Crash Magic for analysis. The following is a list of fields used by the P_Default calculated fields, the field data type, and a description of what the field is used for. These fields are expected by Crash Magic. Removing or adding fields to the P_Default calculated fields will cause problems. An expression is created for each field that will tell Crash Magic how the field is to be calculated. Each expression is based on the fields returned from the query assigned to the calculated field.

_CaseID(string)

Unique identifier for each collision. This field is required by the program.

Example expression:

`AccidentID`

In this example the query field AccidentID would be assigned to the _CaseID calculated field.

_PrimaryStreet(string)

The primary street that the collision occurred on. A valid expression is required by Intersection studies. This is generally the primary street field name of the Intersection study. Collision data that does not use street and cross street to locate collisions should set this field to a blank string.

_CrossStreet(string)

The nearest cross street to the collision. A valid expression is required for use by Intersection studies. This field may be blank if collisions in the database do not have cross street data. This is generally the cross street field name of the Intersection study. Collision data that does not use street and cross street to locate collisions should set this field to a blank string.

_Date(date)

An expression that represents the date the collision occurred. Crash Magic will use this field to find the date a collision occurred.

Example expression:

```
IF(ISNULL(Accidenttime),BuildDate(Accidentdate,0,0,0),BuildDate(Accidentdate,Accidenttime))
```

In this example the _Date field is created from a date and time field from the query. If the time field is null a time of 12:00 AM is added to prevent the BuildDate function from creating a null date value for an invalid time. Adding a valid time to the date field is no longer required by the program, but is recommended for display of the field to users. Users accessing this field should be informed that the _Date field can contain the bad time data. A null date will result in null data for the field.

_Time(date)

An expression that represents the time the collision occurred. Crash Magic will use this field to find the time a collision occurred.

Example:

```
If(ISNULL(Accidentdate),BuildDate(ASDATE("01/01/1900"),Accidenttime),BuildDate(Accidentdate,Accidenttime))
```

In this example the _Time field is created from a date and time field from the query. If the date field is null a date of 1/1/1900 is added to prevent the BuildDate function from creating a null time value for an invalid date. If Accidenttime is null the entire field will be null. Adding a valid date to the time field is not required by the program, but is recommended for display of the field to users. Users accessing this field should be informed that the _Time field can contain the bad date. A null time will result in null data for the field.

_Veh1Dir(number)

Expression that maps vehicle one direction values to the internal directions stored in the .Master configuration. A valid expression is required by the collision diagrams.

Example:

```
Case( Travel_Dir1, "N", _North, "S", _South, "E", _East, "W", _West, _NoDir )
```

Item	Description
Case 84	Case Function

Travel_Dir1	Parameter to be examined (Field name from the study query)
N	Possible value in Travel_Dir1
_North	Crash Magic Master .shared value for North
S	Possible value in Travel_Dir1
_South	Crash Magic Master .shared value for South
E	Possible value in Travel_Dir1
_East	Crash Magic Master .shared value for East
W	Possible value in
_West	Crash Magic Master .shared value for West
_NoDir	Crash Magic Master .shared value for unknown direction

Veh2Dir(number)

Expression that maps vehicle two direction values to the internal directions stored in the .Master configuration. A valid expression is required by the collision diagrams.

Veh3Dir(number)

Expression that maps vehicle three direction values to the internal directions stored in the .Master configuration. A valid expression is recommended for the additional vehicle indicator symbol.

DistFromInt(number)

Distance from the collision to the location reference point or nearest cross street. Distance from intersection allows Crash Magic to determine the correct area for collision placement in some diagram schematics. Pd' Programming recommends this value default to 0 if the field is null for correct collision diagram displays.

DirFromInt(number)

Expression that maps direction from the collision to the location reference point or nearest cross street.

IsSideswipe(boolean)

Expression to determine if the collision was a sideswipe. This is used by collision diagram to determine if the vehicle symbols should be placed side by side.

IsRearEnd(boolean)

Expression to determine if a collision was a rear end. This is used by collision diagrams to determine if one vehicle should be placed behind another.

IsFatality(boolean)

Expression to determine if a fatality was caused by the collision.

IsInjury(boolean)

Expression to determine if an injury was caused by the collision.

_Veh1AheadRank(Number)

Expression to rank vehicle one's position in a collision diagram based on the vehicle maneuver in a rear end collision. This expression is required by collision diagrams.

The following is list of standard ranks based on vehicle movement:

Ahead Rank	Vehicle Movement
100	Backing
90	Parking Maneuver
90	Leaving Parking Position
80	Stopped For Sign/Signal
80	Stopped
75	Legally Parked
75	Illegally Parked
70	Entering Parking Position
65	Slowing/Stopping
60	U Turn
55	Left Turn
55	Left on Red
50	Turning on Red
50	Right Turn
50	Right on Red
50	Other Unsafe Turn
45	Ran off Road Right
45	Ran off Road Left
45	Ran off Road
20	Straight
15	Overtaking/Passing
5	Working on or Pushing Veh
1	Overturn/Rollover
1	Airborne
0	Wrong Way
0	Unknown
0	Unattended
0	Skidding/Control Loss
0	Other
0	Leaving Traffic Lane
0	Leaving Alley/Driveway
0	Jackknife
0	Immersion
0	Fire/Explosion
0	Entering Traffic Lane (merging)
0	Entering Alley/Driveway
0	Driverless Moving Veh
0	Crossing Road
0	Crossed Median
0	Crossed Into Opp Lane
0	Crossed Centerline

0	Changing Lanes
0	Avoiding Veh or Obj

_Veh2AheadRank(Number)

Expression to rank vehicle two's position in a collision diagram based on the vehicle maneuver in a rear end collision. This expression is required by collision diagrams.

_Veh3AheadRank(Number)

Expression to rank vehicle three's position in a collision diagram based on the vehicle maneuver in a rear end collision.

_Veh1RightRank(Number)

Expression to rank vehicle one's position in a collision diagram based on the vehicle maneuver in a sideswipe collision.

_Veh2RightRank(Number)

Expression to rank vehicle Two's position in a collision diagram based on the vehicle maneuver in a sideswipe collision.

_Veh3RightRank(Number)

Expression to rank vehicle Three's position in a collision diagram based on the vehicle maneuver in a sideswipe collision.

_NorthOfInt(boolean)

This expression is used to determine if a collision was North of an intersection.

_SouthOfInt(boolean)

This expression is used to determine if a collision was South of an intersection.

_EastOfInt(boolean)

This expression is used to determine if a collision was East of an intersection.

_WestOfInt(boolean)

This expression is used to determine if a collision was West of an intersection.

_NBArriving(boolean)

This expression is used by collision diagrams to determine placement of collisions within a diagram schematic. This field is required by all standard collision diagrams that are included in the .master shared user. The default expression is:

```
( AllMatch( _North, _Veh1Dir, _Veh2Dir ) .OR. SetMatch( _North, _NoDir, _Veh1Dir,
_Veh2Dir ) ) .AND. ( _DirFromInt .NE. _North )
```

_NBDeparting(boolean)

This expression is used by collision diagrams to determine placement of collisions within a diagram schematic. This field is required by all standard collision diagrams that are included in the .master shared user. The default expression is:

```
( AllMatch( _North, _Veh1Dir, _Veh2Dir ) .OR. SetMatch( _North, _NoDir, _Veh1Dir,
_Veh2Dir ) ) .AND. ( _DirFromInt = _North )
```

_SBArriving(boolean)

This expression is used by collision diagrams to determine placement of collisions within a diagram schematic. This field is required by all standard collision diagrams that are included in the .master shared user. The default expression is:

```
( AllMatch( _South, _Veh1Dir, _Veh2Dir ) .OR. SetMatch( _South, _NoDir, _Veh1Dir,
_Veh2Dir ) ) .AND. ( _DirFromInt .NE. _South )
```

_SBDeparting(boolean)

This expression is used by collision diagrams to determine placement of collisions within a diagram schematic. This field is required by all standard collision diagrams that are included in the .master shared user. The default expression is:

```
( AllMatch( _South, _Veh1Dir, _Veh2Dir ) .OR. SetMatch( _South, _NoDir, _Veh1Dir,
_Veh2Dir ) ) .AND. ( _DirFromInt = _South )
```

_EBArriving(boolean)

This expression is used by collision diagrams to determine placement of collisions within a diagram schematic. This field is required by all standard collision diagrams that are included in the .master shared user. The default expression is:

```
( AllMatch( _East, _Veh1Dir, _Veh2Dir ) .OR. SetMatch( _East, _NoDir, _Veh1Dir,
_Veh2Dir ) ) .AND. ( _DirFromInt .NE. _East )
```

_EBDeparting(boolean)

This expression is used by collision diagrams to determine placement of collisions within a diagram schematic. This field is required by all standard collision diagrams that are included in the .master shared user. The default expression is:

```
( AllMatch( _East, _Veh1Dir, _Veh2Dir ) .OR. SetMatch( _East, _NoDir, _Veh1Dir,
_Veh2Dir ) ) .AND. ( _DirFromInt = _East )
```

_WBArriving(boolean)

This expression is used by collision diagrams to determine placement of collisions within a diagram schematic. This field is required by all standard collision diagrams that are included in the .master shared user. The default expression is:

```
( AllMatch( _West, _Veh1Dir, _Veh2Dir ) .OR. SetMatch( _West, _NoDir, _Veh1Dir,
_Veh2Dir ) ) .AND. ( _DirFromInt .NE. _West )
```

_WBDeparting(boolean)

This expression is used by collision diagrams to determine placement of collisions within a diagram schematic. This field is required by all standard collision diagrams that are included in the .master shared user. The default expression is:

```
( AllMatch( _West, _Veh1Dir, _Veh2Dir ) .OR. SetMatch( _West, _NoDir, _Veh1Dir,
_Veh2Dir ) ) .AND. ( _DirFromInt .EQ. _West )
```

_NEBArriving(boolean)

This expression is used by collision diagrams to determine placement of collisions within a diagram schematic. This field is required by diagrams that support non-cardinal directions. The vast majority of collision data does not support non-cardinal directions, and in most cases is set to false.

_NEBDeparting(boolean)

This expression is used by collision diagrams to determine placement of collisions within a diagram schematic. This field is required by diagrams that support non-cardinal directions. The vast majority of collision data does not support non-cardinal directions, and in most cases is set to false.

_NWBArriving(boolean)

This expression is used by collision diagrams to determine placement of collisions within a diagram schematic. This field is required by diagrams that support non-cardinal directions. The vast majority of collision data does not support non-cardinal directions, and in most cases is set to false.

_NWBDeparting(boolean)

This expression is used by collision diagrams to determine placement of collisions within a diagram schematic. This field is required by diagrams that support non-cardinal directions. The vast majority of collision data does not support non-cardinal directions, and in most cases is set to false.

_SEBArriving(boolean)

This expression is used by collision diagrams to determine placement of collisions within a diagram schematic. This field is required by diagrams that support non-cardinal directions. The vast majority of collision data does not support non-cardinal directions, and in most cases is set to false.

_SEBDeparting(boolean)

This expression is used by collision diagrams to determine placement of collisions within a diagram schematic. This field is required by diagrams that support non-cardinal directions. The vast majority of collision data does not support non-cardinal directions, and in most cases is set to false.

_SWBArriving(boolean)

This expression is used by collision diagrams to determine placement of collisions within a diagram schematic. This field is required by diagrams that support non-cardinal directions. The vast majority of collision data does not support non-cardinal directions, and in most cases is set to false.

_SWBDeparting(boolean)

This expression is used by collision diagrams to determine placement of collisions within a diagram schematic. This field is required by diagrams that support non-cardinal directions. The vast majority of collision data does not support non-cardinal directions, and in most cases is set to false.

_UniqueFields(string)

A string expression that is used to return parameters to the [ClickOn](#)²⁷³ query. This field is required for the ClickOn report in collision diagrams. The expression built

from the _UniqueFields is placed in a diagram for each collision. This allows the correct parameters to be passed to crash Magic from collision diagrams when a collision is selected.

The Format is <Parameter>:<Datatype>=

<Expression>[;<Parameter>:<Datatype>= <Expression>][...]

Parameter is the parameter required by the ClickOn query, Datatype is the datatype of the parameter required by the ClickOn query, and Expression is the

Example expression:

```
"CRASH_NUM:integer=" + String(CASE_NUM,0) + ";CRASH_DATE:Date=" + CRASH_DATE
```

Parameter	Description
CRASH_NUM	The name of a parameter that will be used by the ClickOn query.
:integer	This parameter is the datatype required by the ClickOn query. The result of the _UniqueFields expression are passed as string values, and this parameter tells Crash Magic what the datatype should be converted to for use in the ClickOn.
=	Used to set the parameter value in the diagram.
+	This operator is used to join the strings together.
String(CASE_NUM,0)	This is the actual value that will be passed to the ClickOn query. In this example CASE_NUM is an integer field in the collision database. The string function is used to convert the integer value to a string to concatenate the string values that are embedded in the collision diagram.
;	This value will separate the parameters being passed when the field is inserted into the diagram.
CRASH_DATE	The name of another parameter that will be used by the ClickOn query.
:DATE	This parameter is the datatype required by the ClickOn query. The result of the _UniqueFields expression are passed as string values, and this parameter tells Crash Magic what the datatype should be converted to for use in the ClickOn.
CRASH_DATE	This is the actual value that will be passed to the ClickOn query. No explicit conversion is required.

Result from the following expression with a value of 503871 for CRASH_NUM and a CRASH_DATE of 01/01/2005 return the following:

CRASH_NUM:integer=503871;CRASH_DATE:Date=01/01/2005

_Veh1DirIsVeh2Dir(boolean)

Expression to determine if vehicle one's direction of travel should be the same as vehicle two's direction of travel.

_Veh2DirIsVeh1Dir(boolean)

Expression to determine if vehicle two's direction of travel should be the same as vehicle one's direction of travel.

_Milepost(Number)

Expression to determine the mile post that a collision occurred at. This is used by milepost corridor schematics to place collisions in the correct schematic mile. This should be set to zero if the collision data does not contain mile post data.

_BlockAddress(Number)

Expression to determine the block address of a collision. This is used by address corridor schematics to place collisions in the correct schematic block. This should be set to zero if the collision data does not contain an block address data.

The U_UserSpecific calculated fields are for calculated fields that are specific to the User Groups data. These fields are not required by the program. User specific fields can simplify the creation of an object map, and allow users to add fields in their reports with complex calculations.

This is the last calculated field to be loaded by Crash Magic, and as a result can use any calculated field group that has already been loaded.

The following are some example user specific fields. Your fields will be different:

SampleReport (string)

```
BinaryFileContent("SampleReports", "Report_" + _CASEID + ".jpg")
```

This is an example of an image file. This will allow users to select the SampleReport field in the field list editor of the [ClickOn](#)^[20] report to display an image. The [BinaryFileContent](#)^[84] function is used to define the image file. The image file name is calculated based on the Case Id of the collision selected. For example if a user clicks on Case Id 58214 in a collision diagram then Crash Magic would try to retrieve the file Report_58214.jpg for display.

VehSameDir (boolean)

```
setMatch(_Veh1Dir,_North,_Veh2Dir,_North)
.OR. setMatch(_Veh1Dir,_South,_Veh2Dir,_South)
.OR. setMatch(_Veh1Dir,_East,_Veh2Dir,_East)
.OR. setMatch(_Veh1Dir,_West,_Veh2Dir,_West)
```

This example code uses the setMatch function to determine if vehicle one and vehicle two are traveling in the same direction.

IsOverturn (boolean)

```
(EventCode = 7)
```

In this example if the query field EventCode is equal to 7, then the collision is resulted in an overturned vehicle.

Veh1IsPed (boolean)

```
UnitTypeCode_one = 5
```

In this example if the query field UnitTypeCode_one is equal to 5 then vehicle one is a pedestrian.

AvgCollisionCost (integer)

```
If(_IsFatality,
    500000,
    If(_IsInjury,
        100000,
        5000))
```

This example uses the calculated fields IsFatality and IsInjury from the P_Default calculated fields group to assign a numeric value to each type of collision. This can be used to derive an average cost for each type of collision.

Configuration Helper (confighelper)

The Configuration Helper is used to store data about the tables and fields that Crash Magic will use and import. A Configuration Helper can generate sql scripts and objects used in a configuration. In most cases we will be copying the format of crash data from a client file or database. This process shows how to create a Configuration Helper from a connection to a text file in a directory with a schema.ini file. Connecting to a ASCII data source requires that you have already defined a [schema.ini](#) file from the previous section and installed Microsoft Jet 4.0 OLE provider.

Double click the Pd Data Structure Exporter(PdDataStructureExporter.exe).

Select the driver to connect to your data files:

1. Click on the ... button to define a connection string to your data source.
2. Click the ... button to build a connection string.
3. Click the Build... button.
4. Select the Microsoft Jet 4.0 OLE provider.
5. Click the next button.

Select the data source directory:

1. With the Data Link Properties open Click on the All tab.
2. In the list of properties click on the Data Source under the Name column.
3. Click the Edit Value... button.
4. Enter the directory path where the schema.ini file is located(Example location C:\TrafficRecords\ImportFiles) in the Property Value box.
5. Click the OK button to close the Edit Property Value window.

Select the Extended Properties:

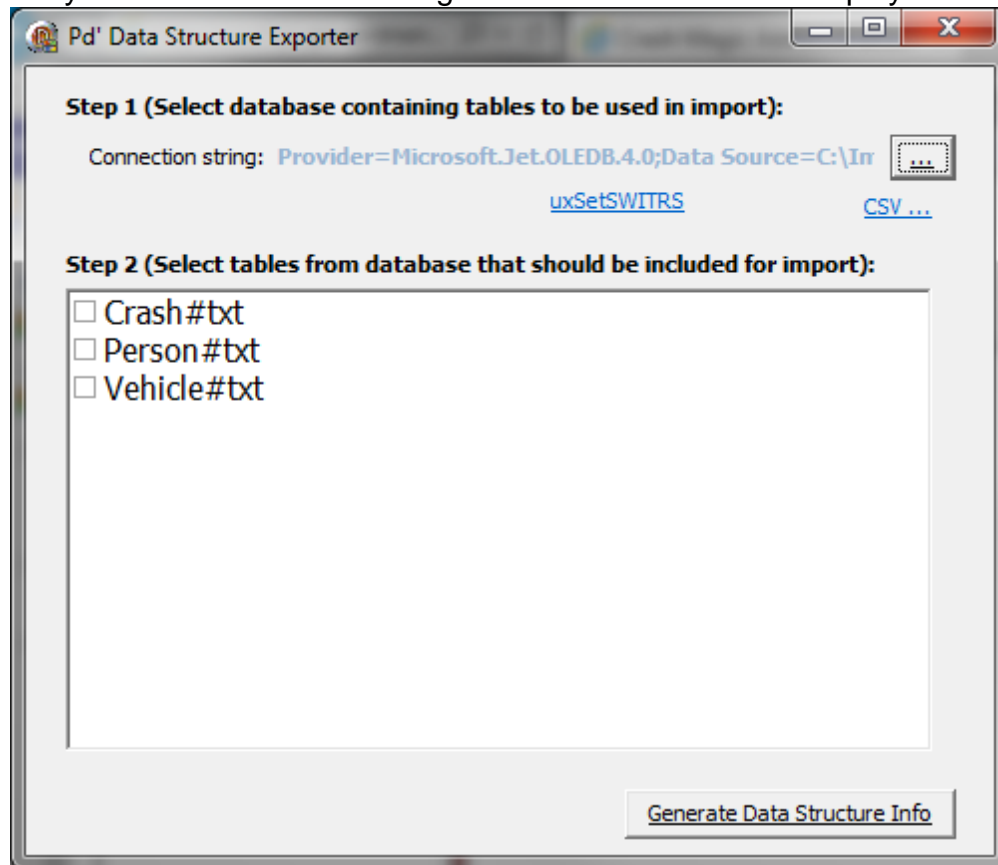
1. With the All tab of the Data Link Properties window open click on the Extended Properties under the Name column.
2. Click the Edit Value... button.
3. Enter text in the property value box.
4. Click OK to close the Edit Property Value window.

Close the windows and test the connection string:

1. Click Ok to close the Data Link Properties window.
2. Click OK to close the ConnectionSettingForm. ConnectionString window.
3. Click the Test button on the Edit connection settings window. A successful test will display the ADO version information. An unsuccessful test will display connection failed(Check that all of the steps and requirements have been followed).
4. Click OK to close the edit connection settings window.

Generate the configuration helper XML:

1. Once you have a connection string the a list of tables will be displayed under the step



2. Check the boxes next to each table(Your list of tables will be different).
3. Click the Generate Data Structure Info button to open a text file with the Configuration Helper XML.
4. Click the Edit drop down menu.
5. Click Select All.
6. Click the Edit drop down menu.
7. Click Copy.

Create a Configuration Helper in Crash Magic:

1. Log into Crash Magic as the Group Administrator.
2. Click on the .shared user under the user group you have logged into.
3. While on the green Settings table of the .shared user Click the drop down menu under Utility functions.
4. Select Configuration Helper (confighelper) in the drop down menu.
5. Click the Create new PSRattr button.

Add the XML to the helper:

1. While on the configuration helper that was created from the previous task click on the Raw XML (advanced) tab.
2. Right click in the white area displaying the XML.
3. Click on Select all from the right click menu.
4. Right click again in xml the area.
5. Click paste to paste the XML from the Generate the configuration helper XML task.

The DB to XML PSRattr defines how Crash Magic will create an xml data file for import from the data source used. The following steps show how to create a DB to XML PSRattr from the Configuration Helper.

Locate the Configuration Helper:

1. Log into Crash Magic as the Group Administrator.
2. Click on the .shared user under the user group you have logged into.
3. Click the + sign next to the .shared user to display the attributes for the user.
4. Click on the + sign next Configuration Helpers.
5. Click on the greet settings tab.

Generate a DB to XML PSRattr definition:

1. While on the settings tab of the Configuration Helper Check the box next to DB to XML.
2. Enter a name for this import definition in the New attributes name box.
3. Click the Create attributes button to create a an import definition.

This will generate the DB to XML in the tree.

The Configuration Helper is able to create a basic DB to XML structure but if you have more than one table defined you will need to tell Crash Magic how each table queried will relate when the XML file is written. The unique fields, [schema.ini](#)²⁸⁶ and parent child relationships will need to be defined.

The following steps assume that you are logged into the configuration as Group Administrator and are on the Raw XML (advanced) tab of the

The following is a sample DB to XML

```
<pdroot>
  <DbToXml>
    <General>
      <OutputDir/>
    </General>
    <TableSpecs>
      <TableSpec>
        <Name>Crashes</Name>
        <Primary>True</Primary>
        <ConnectionString/>
        <Login/>
        <Password/>
        <BaseName>Crashe</BaseName>
        <GroupName/>
        <ItemName/>
        <UniqueFields/>
        <QueryLines>
```

Configuration notes (configurationnotes)


Configuration notes is a place to store information about decisions, problems, exceptions unique to this configuration. This becomes critical information long after the configuration is completed when questions arise about program functionality and output. This document is also presented to the client upon delivery.

There is no configuration notes editor, just XML in a text editor. The format follows:

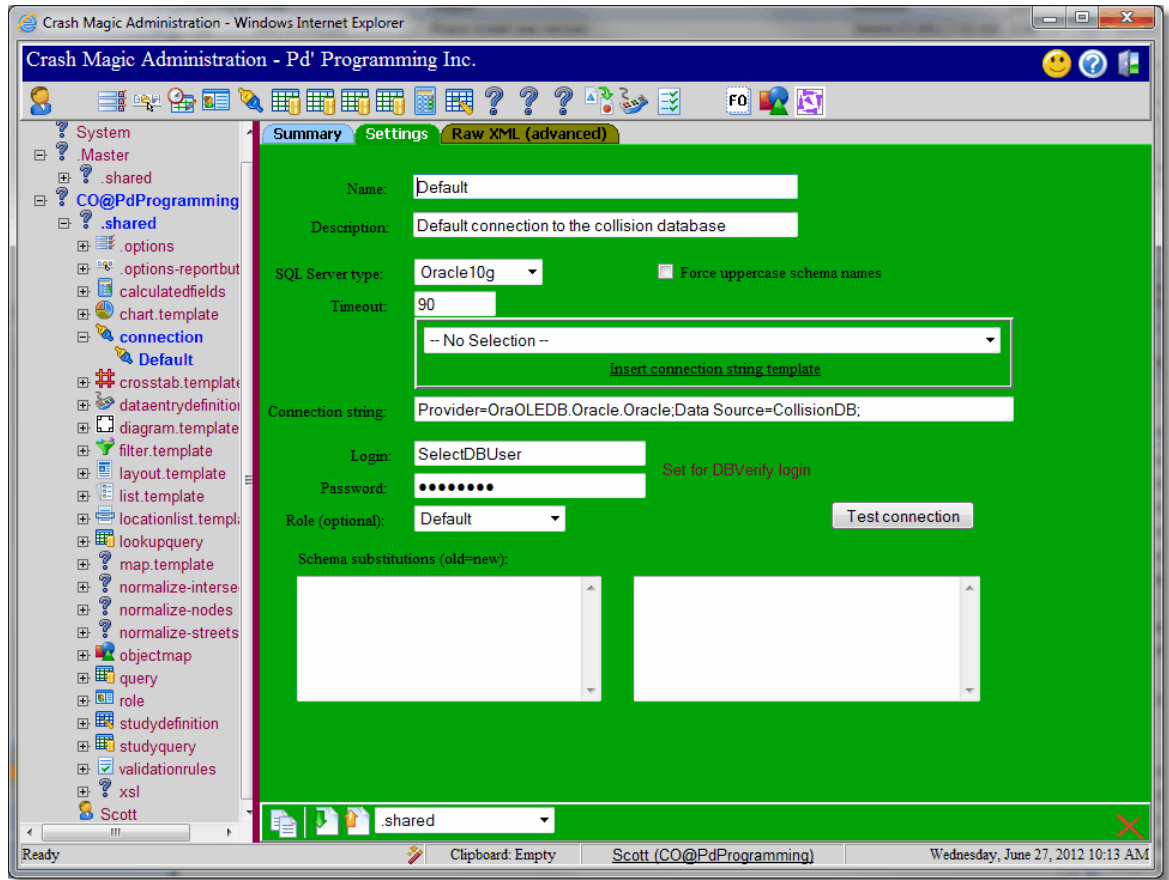
```
<Notes>
  <Note>
    <Author> </Author>
    <NoteDate> </NoteDate>
    <Subject> </Subject>
    <Lines>
      <Line> </Line>
    </Lines>
  </Note>
</Notes>
```

Connection (connection)

Creating a connection assumes that the appropriate database drivers have been installed on the server hosting the Crash Magic application.

To Create a connection click the connection button  while under the .shared user of the group. This first connection will be a read only connection to the database for collision analysis. The information for this connection should be provided by your database administrator. The Settings panel requires information in the following fields.

Other connections may be required (for example a connection for data entry), but in this example we will focus on creating a connection only for analysis of collision data.



Name: Enter the name you will call the connection being created.

Description: Enter a description of what the connection is used for.

SQL Server type: Select the database platform that Crash Magic will be connecting to. See your DBA if you are unsure.

Timeout: The number of seconds a query will be allowed to run. Crash Magic display an error to the user once this time has elapsed. It is recommended that this be left at the default of 30 seconds. This can be raised if a larger query time is required.

Force uppercase schema names: Check this box if your database uses upper case schema names. This forces Crash Magic to use upper case schema names (Most users will leave this unchecked).

Provider drop down selection: Select a provider from the drop down list of providers that can be selected to building a connection string. Try to select from the Pd' Default connection String section. Contact your database administrator if you are unsure which template to select

Insert connection string template: Click this link to insert a new connection sting. This will add a connection string template to the Connection string field based on the provider selected in the drop down menu.

Connection string: Change the connection string parameters to access your database(In most cases this will mean changing the Data Source=(Enter the name of your data source) and if required the Initial Catalog=(Enter the name of the catalog) Examples

Oracle

Sample connection string provided:

```
Provider=OraOLEDB.Oracle;Data Source=MyOracleDB;
```

MyOracleDB should be changed to the data source specified in the Oracle tns names file. In the example below the CrashMagic is the specified database alias in the Oracle tns names file.

```
Provider=OraOLEDB.Oracle;Data Source=CrashMagic;
```

MS SQL Server

Sample connection string provided:

```
Provider=sqloledb;Data Source=myServerAddress;Initial Catalog=myDataBase;
```

myServerAddress would be changed to the name of the server hosting the database, and myDatabase would be changed to the name of the database used on the server. In the example below SQLDbServer is the name of the host and CrashMagic is the database on the server.

```
Provider=sqloledb;Data Source=SQLDbServer;Initial Catalog=CrashMagic;
```

DB2

Sample connection string provided

```
Provider=IBMDADB2;Database=myDataBase;Hostname=myServerAddress;Protocol=TCPIP; Port=50000;
```

myDatabase is the name of the DB2 database and Hostname is the name of the server where the database resides. In the example below CrashM is the database name and DB2Server is the name if the server where CrashM resides.

```
Provider=IBMDADB2;Database=CrashM;Hostname=DB2Server;Protocol=TCPIP; Port=50000;
```

Set for DBVerify login: Setting the login and password depends on the collision database being connected to. If each Crash Magic user will have their own database account to access the collision data then click the "Set for DBVerify login". This will populated the login field with %USER.LOGIN%. Leave password field blank as the user will provide this when logging into Crash Magic.

Login: If not using DBVerify enter the login name to the collision database that Crash Magic will use.

Password: If not using DBVerify enter the password for the collision database login.

Role: Click the drop down menu and select the name of the [role](#)²⁷⁹ created from the previous task. If no roles have been defined in Crash Magic you will need to do so before the connection can be used.

SQL Join Syntax: This is the SQL syntax that the database is using. (Most users should select Ansi99)

Test connection: This button will allow you to test the connection to the database. If the connection succeeds a window will open informing you that the connection has succeeded, and provide information on the database drivers in use.

Schema substitutions(old=new): This feature has not been implemented.

Data Entry Definition (dataentrydefinition)

The Data Entry definition is used to define a data entry form. This form is used to map entry fields to a client database.

- The top section of the editor is simply the name and description of Data Entry Definition. This section also contains a list of possible validation rules to choose from.
- The next section "Update locations" contains a list of update locations the properties that can be set for each location
- The bottom section contains a list of the entry fields. When the "Entry Fields" checkbox is checked, the list reflects all the fields that will be displayed on the data entry form for editing. When the "Unique Fields" checkbox is checked, the list of fields are replaced with the unique fields for the currently selected update location.

Sample data entry form. Update locations and fields will be different for each client.

Name - Name of the Data Entry definition. The main update location, that is the table with a one record per crash, that other tables hang off of. (usually crash, incident, environment, etc.) must be titled "Main".

Description - Description of the data entry definition.

Validation Rules - Specifies the name of the [validation rules](#)^[289] that will be used by the data entry form.

Viewer EXE - Expression that resolves to the name of the local program that can be used to open <Viewer File>. If this value is left blank, the windows system default viewer for <Viewer File> will be used. This field is intended for use when the default viewer is not the desired viewer, or additional parameters are desired.

Viewer File - Expression that resolves to the name of the scanned image / hardcopy for the crash. When loading a new record for editing, this value will be evaluated. If the resulting folder + file name exists on the disk, it will be opened for viewing along with the record. In some cases, when the expression can not be fully evaluated, the viewer will not be launched. (i.e. if the folder requires knowing the crash date, and the date hasn't been entered yet) In this situation, there is a "!" button on the data entry form. Pressing that button after entering the required data will re-evaluate the Viewer File and attempt to open the viewer again.
Some example expressions:

```
"c:\CrashRecords\" + AsString(Year(FormCrashDate.value),0 ) + "\" + CrashId.Value + ".pdf"
"\\Server12\CrashData\" + AsString(Year(FormCrashDate.value),0 ) + "\" + CrashId.Value + ".pdf"
"\\Server12\CrashData\" + CrashId.Value + ".tif"
```

The field names used in the examples above are specific to the data entry definition.

Update Location - This section specifies the tables that will be updated from the data entry form. A single collision record may create records in multiple tables. Each data entry form field contains an update location that specifies the table. A table can be specified multiple times. For example a single crash record may require several vehicle records be added to a vehicle table. One update location would then be added for each vehicle displayed on the data entry form.

Name - A name used to identify the update location. Each field displayed on a data entry form is tied to an update location.

Table Name - The schema and name of the table to be updated by the selected Update Location.

Connection - The name of a Crash Magic Online [connection](#)²²⁷ that will be used by the data entry form. This connection must have insert, update and delete privileges on all tables defined in the update locations.

Do Write - A Boolean expression that defines when the table specified by the update location will be updated. In this example the expression is (IsNull(FormUnit2Type.Value) .EQ. FALSE). This means that the table defined in Update location Unit2 will be updated when the form field Unit 2 Type is not blank. This field uses Crash Magic filter [functions](#)⁸⁰. The expression <Field Name>.Value is used to access a value entered in a form field.


Entry Fields - Allows editing of fields show in the data entry form.

Unique Fields - Allows the editing of unique fields for each Update location. Unique fields identify a unique record for an update location. For example a vehicle record may need the case Id and the vehicle number to identify the record as unique. These fields will be displayed in the data entry New Record and Edit Record tabs. In most case only the unique fields of the first update location will need to be visible. For example if the main crash record requires a case id field, and the vehicle records require the same case id number to tie the vehicle records to the crash record. In this case only the case id field for the main crash record should be displayed, and the case id field for the vehicle records will be hidden, but use the case id from the crash record would be specified in the expression for the fields.

Field Editor



- Add an entry field.

 - Remove an entry field.

 - Move an entry field up in the list.

 - Move an entry field down in the list.

Entry Field Name - Name that will be used to identify the field on the form.

Visible - Determines if the field and caption for that field should be displayed on the data entry form.

Control Type - The type of field and prompt the field will provide. For example the Date prompt type will display a calendar selection prompt. The following is a list of possible prompt types.

- String - A text value
- WholeNumber - A numeric value with not decimal places.
- DecimalNumber -
- Date- Date
- Time - Time field
- Lookup - For fields that have lookup values.
- IndexedPrimartStreet - Primary street. This control validates the street name entered.
- IndexedCrossStreet - Nearest cross street prompt.
- NodeNum - Node ID
- LargeLookup -
- NextID -
- YesNo - Yes No check box fields.
- Label - For display of a field as a label.
- Annotation - Causes only the caption to display.
- Image - Display a rasterimage
- Memo - A multi-line edit control

Location (X, Y) - Sets the location of the field from the top left corner of the form. An omitted or 0 X value will cause the field to be placed below the previous Entry Field. An omitted or 0 value in the X and Y fields will cause the field to be placed below and aligned with the previous field.

Properties -Defines the properties for a field. The following is a list of properties for different types of fields.

Lookup fields - Defines a list of values to be displayed for a field.

- Lookups=<[DBFIELD](#)²⁷⁵> - Defines the lookup descriptions that will be displayed for the data entry field. Example Lookups=Weather would display the lookups for the Weather field.
- LookupFormat=<Format> - Defines how the field should be displayed. Example LookupFormat=Value+Lookup - This will display the raw data value and the human readable value.
- LookupSort=<Sort> - Defines the order that looked up values will be displayed. Example LookupSort=NumberValue will display the list of values ordered by the raw data number.

```
Properties: Lookups=Lightingcondition
            LookupFormat=Value+Lookup
            LookupSort=NumberValue
```

Example lookup for lightingcondition

Primary street field - Validates street names against the current streets.

- IntersectionNormalizer=<[Intersection Normalizer Name](#)²⁶⁴> - This property restricts the field to street names returned from an intersection normalizer. This property is used to validate streets in data entry. Example IntersectionNormalizer=MagicCity would restrict the field to only streets defined in the intersection normalizer named MagicCity.

Cross Street field - Provides a list

- IntersectionNormalizer=<[Intersection Normalizer Name](#)²⁶⁴> - This property restricts the field to street names returned from an intersection normalizer. This property is used to validate streets in data entry. Example IntersectionNormalizer=MagicCity would restrict the field to only streets defined in the intersection normalizer named MagicCity.
- RelatedPrimaryStreet=<Field name> - When used in conjunction with IntersectionNormalizer this property will limit a cross street field to only those defined by the IntersectionNormalizer and street name a user has selected. Example:

```
IntersectionNormalizer=MagicCity
RelatedPrimaryStreet=PrimaryStreet
```

In this example if a user entered Broadway into the PrimaryStreet form field, and then selected a field with the example properties, the data entry form would display the cross streets to Broadway as defined in the MagicCity normalizer.

Node fields - Node fields can select a node Id based on a street and cross street entered in previous fields, and display a map of the area.

- LookupQuery=<Query name> - Names the [lookup](#)²⁷⁵ query for nodes.
- Lookups=<[DBFIELD](#)²⁷⁵> - Defines the lookup descriptions that will be displayed for the data entry field.
- LookupFormat=<Format> - Defines how the field should be displayed. Example LookupFormat=Value+Lookup - This will display the raw data value and the name of the node.
- LookupSort=<Sort> - Defines the order that looked up values will be displayed.

- **RelatedPrimaryStreet=<Field Name>** - Name of the primary street field that will be used to search for the node.
- **RelatedCrossStreet=<Field Name>** - Name of the cross street field that will be used to search for the node.
- **Map=**
- **MapNodeZoom=**
- **NodeLayer=**
- **NodeIDField=<DBVALUE>** - Field from the lookup query that uniquely identifies each node.

Horiz. Anchor - The horizontal anchor point of the field. When left is specified the top left side of the field will be the Location (X, Y) coordinates. When center is specified the top center of the field will be the Location (X, Y) coordinates.

Width - Sets the width of the entry field. If left blank or 0 the field width will set to the default width for the control type specified.

Caption - This is the prompt that will be displayed next to the field on the form based on the Prompt settings position.

Position - Sets the position of the caption relative to the field. Selecting a value of None will hide the caption.

Value Settings - Sets the position that a looked up value will be displayed relative to the entry field.

Update Location - Sets the update location of the form field.

Read Only - When checked any value entered will not be written back to the database. This is used for labels and when the form field will be used to populate another form field.

Field name - The name of the database field that will be updated. This field name must be in the table of the update location.

Field Type - The database field data type.

Default - An expression that populates an empty field with a default value when the form displays.

Expression - An expression that sets the value of a database field when the record is saved. Example

```
BuildDate(AccidentFormDate.Value, AccidentFormTime.Value)
```

This expression can be used to populate a single datetime field in a database by taking the date and time field from a data entry form.

Note: At the time of this writing, the program did not support adding update locations. This set of instructions assumes this is still the case. If the data entry definition contains a button to add an update location, please use that method instead.

Before beginning, you will need the name of the streets table and the field names to refer to.

1. Under .config, create the psrattr Data Entry Definition. Name it Streets.
2. Go to the .xml tab and paste this in:

```
<pdroot>
  <DataEntryDefinition>
    <General>
      <AsLoadedChecksum>0</AsLoadedChecksum>
      <Notes/>
      <ValidationRules></ValidationRules>
    </General>
    <UpdateLocs>
      <UpdateLoc>
        <Name>Main</Name>
        <TABLENAME>DBO.STREETS</TABLENAME>
        <Connection>Writer</Connection>
        <LoadedCount>0</LoadedCount>
        <DoWrite>False</DoWrite>
        <DoWriteExpression>true</DoWriteExpression>
        <UniqueFields>
          <EntryField>
            <Name>StreetNum</Name>
            <XPos>0</XPos>
            <YPos>0</YPos>
            <Width>100</Width>
            <EntryWidth>0</EntryWidth>
            <ValuePos>None</ValuePos>
            <HAnchor>Left</HAnchor>
            <JustifyLabel>False</JustifyLabel>
            <LabelPos>Left</LabelPos>
            <ValueWidth>300</ValueWidth>
            <ContentMax>2147483647</ContentMax>
            <ContentMin>-2147483646</ContentMin>
            <UpdateLocName>Main</UpdateLocName>
            <FieldName>StreetNum</FieldName>
            <FieldType>Integer</FieldType>
            <PromptType>WholeNumber</PromptType>
            <PromptText>Street Number:</PromptText>
            <DBValue/>
```



```

<Visible>True</Visible>
<OnlyRead>False</OnlyRead>
<Default/>
<Expression/>
<CustomProperties>
  <Property Name="IntersectionNormalizer">
    <Value>Default</Value>
  </Property>
</CustomProperties>
</EntryField>
</UniqueFields>
</UpdateLoc>
</UpdateLocs>
<EntryFields>
  <EntryField>
    <Name>LabelStreetNum</Name>
    <XPos>0</XPos>
    <YPos>0</YPos>
    <Width>50</Width>
    <EntryWidth>0</EntryWidth>
    <ValuePos>Right</ValuePos>
    <HAnchor>Left</HAnchor>
    <JustifyLabel>False</JustifyLabel>
    <LabelPos>Left</LabelPos>
    <ValueWidth>300</ValueWidth>
    <ContentMax>2147483647</ContentMax>
    <ContentMin>-2147483646</ContentMin>
    <UpdateLocName>Main</UpdateLocName>
    <FieldName>StreetNum</FieldName>
    <FieldType>Integer</FieldType>
    <PromptType>Label</PromptType>
    <PromptText>Street Number:</PromptText>
    <DBValue/>
    <Visible>True</Visible>
    <OnlyRead>True</OnlyRead>
    <Default/>
    <Expression/>
    <CustomProperties/>
  </EntryField>
  <EntryField>
    <Name>StreetName</Name>
    <XPos>0</XPos>
    <YPos>0</YPos>
    <Width>100</Width>
    <EntryWidth>0</EntryWidth>
    <ValuePos>None</ValuePos>
    <HAnchor>Left</HAnchor>
    <JustifyLabel>False</JustifyLabel>
    <LabelPos>Left</LabelPos>
    <ValueWidth>300</ValueWidth>
    <ContentMax>50</ContentMax>
    <ContentMin>0</ContentMin>
    <UpdateLocName>Main</UpdateLocName>
    <FieldName>StreetName</FieldName>
    <FieldType>String</FieldType>
    <PromptType>String</PromptType>
    <PromptText>Street Name:</PromptText>
    <DBValue/>
    <Visible>True</Visible>
    <OnlyRead>False</OnlyRead>
    <Default/>

```

```

        <Expression/>
        <CustomProperties/>
    </EntryField>
</EntryFields>
</DataEntryDefinition>
</pdroot>

```

3. Change the table name at the top (highlighted in yellow) to match the table name in the database. **DO THIS BEFORE SAVING OR CLICKING ON THE TREE.**
4. Then check the field names to make sure they match what's in the database (also highlighted in yellow).
5. You're done, unless you need another unique field qualifier, like city or county. Then go to the next step.
6. Copy another Data Entry section below, under the Unique Fields section at the top (after Street Num):

```

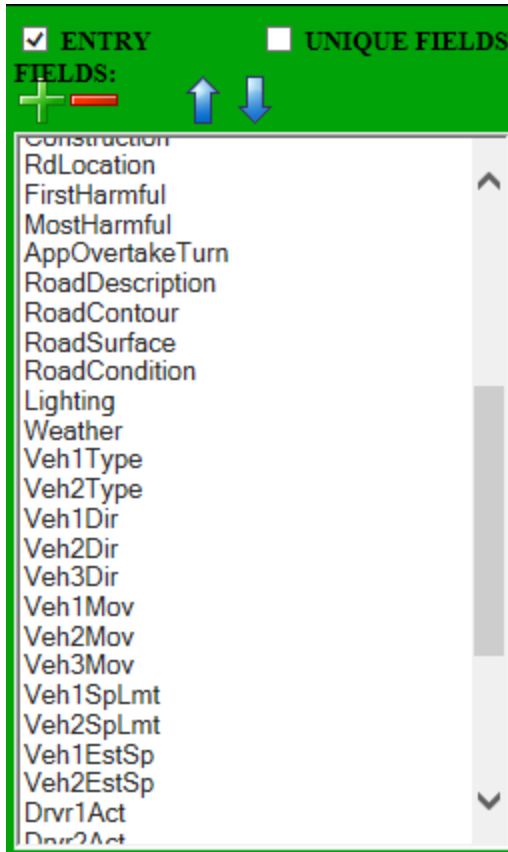
<EntryField>
    <Name>CountyCode</Name>
    <XPos>0</XPos>
    <YPos>0</YPos>
    <Width>50</Width>
    <EntryWidth>0</EntryWidth>
    <ValuePos>Right</ValuePos>
    <HAnchor>Left</HAnchor>
    <JustifyLabel>False</JustifyLabel>
    <LabelPos>Left</LabelPos>
    <ValueWidth>50</ValueWidth>
    <ContentMax>2</ContentMax>
    <ContentMin>0</ContentMin>
    <UpdateLocName>Main</UpdateLocName>
    <FieldName>CNTY_CDE</FieldName>
    <FieldType>WideString</FieldType>
    <PromptType>Lookup</PromptType>
    <PromptText>CountyCode:</PromptText>
    <DBValue/>
    <Visible>True</Visible>
    <OnlyRead>False</OnlyRead>
    <Default/>
    <Expression/>
    <CustomProperties>
        <Property Name="Lookups">
            <Value>CNTY</Value>
        </Property>
        <Property Name="LookupFormat">
            <Value>Value+Lookup</Value>
        </Property>
        <Property Name="LookupSort">
            <Value>NumberValue</Value>
        </Property>
    </CustomProperties>
</EntryField>

```

7. Edit the field names highlighted in yellow above to match the database for the street qualifier field.

First, use the Configuration Helper to generate a basic Data Entry Definition. The important part here is to have it fill the list of fields.

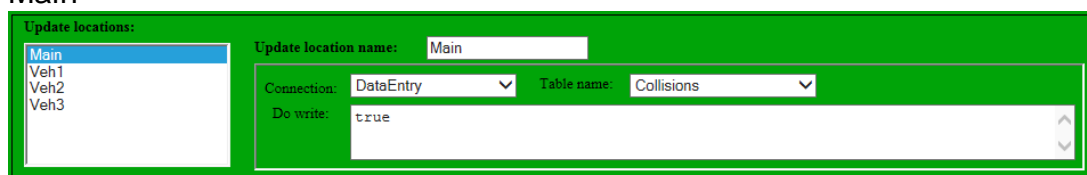
The data entry definition stores data in a relational database, to multiple tables, but it is flat itself. This means that you can't enter one vehicle after another until you're done. The data entry definition may contain entry fields for as many vehicles as you like, but that is exactly how many may be entered. Therefore there need to be fields such as "VehicleDirection1", "VehicleDirection2", "VehicleDirection3", etc.



In order to store a crash record and three vehicles, there need to be four "Update locations". Each field must be assigned to an update location. An update location specifies a connection to a database, a table within that database to update, and a "do write" expression. The do write expression is used to specify whether a particular row should be written. It is frequently used to prevent writing vehicles 2 or 3, or other rows for data that wasn't entered.

For our example, we will create the following update locations:

- Main



This "Main" update location indicates that it will connect with using the "DataEntry"

connection. This, or the "Writer" connection typically provide read/write access the database. This update location writes to the "Collisions" table. Since the Do write expression is "true", saving a record will always result in any fields associated with the "Main" update location being written to the database.

The screenshot shows the 'ENTRY' configuration window for a field named 'CaselD'. The window has a green border and contains several sections:

- ENTRY FIELDS:** A list on the left with 'CaselD' selected.
- UNIQUE FIELDS for [Main]:** A checkbox that is checked.
- sample entry field layout:** A preview showing 'Prompt: Entry' and 'Value'.
- Entry field name:** 'CaselD'.
- Visible:** A checked checkbox.
- Entry settings:**
 - Control type:** 'String'.
 - Location (x,y):** '0' and '0'.
 - Horiz. Anchor:** 'Left'.
 - Width:** '100'.
- Prompt settings:**
 - Caption:** 'Case Id:'.
 - Position:** 'Left'.
- Value settings:**
 - Position:** 'Right'.
- Data settings:**
 - Update location:** 'Main'.
 - Field name:** 'CaselD'.
 - Field type:** 'String'.
 - Default:** (empty).
 - Expression:** (empty).
 - Read only:** A checkbox that is unchecked.

Each update location also has one or more "Unique fields" associated with it. These fields are used to load the record, and must identify it uniquely across all the data in the table that it writes to. Typically, the "main" or "environment" record type will have a single CaselD field for its unique field. Since the unique fields for the main record are used as prompts when creating or loading a new crash record, you will note that CaselD is marked as visible. Its control type is set to string, common for case id's. The other important values here are the data settings where "Main" is specified as the update location; the Field name is "CaselD".

- Veh1

The screenshot shows the 'Update locations' configuration window for a location named 'Veh1'. The window has a green border and contains several sections:

- Update locations:** A list on the left with 'Main', 'Veh1', 'Veh2', and 'Veh3'. 'Veh1' is selected.
- Update location name:** 'Veh1'.
- Connection:** 'DataEntry'.
- Table name:** 'Vehicles'.
- Do write:** A text area containing the expression: `DataCount(Veh1Type.Value, Veh1Dir.Value, Veh1Mov.Value, Veh1SpLmt.Value, Veh1EstSp.Value, Drvr1Act.Value, Drvr1ContrFact.Value, Ped1Act.Value, Veh1Defect.Value, Unit1SafetyEq.Value, Unit1Age.Value) > 0`.

The update location for "Veh1" is much more interesting. As with the "Main" update location, it uses the DataEntry connection. Since it writes to the vehicles table, its table name is "Vehicles". But notice that the Do write expression contains quite a bit of text. This expression is testing to see if any of the fields related to Veh1 have values in them. If any single one of them has a value, (is not null) then the Veh1 record will be written to the database.

☐ ENTRY ☒ UNIQUE FIELDS for [Veh1]

sample entry field layout:
Prompt: Entry Value

ENTRY FIELDS:

- CaseIDVeh1
- VehNum1

Entry field name: CaseIDVeh1 ☐ Visible

Entry settings: [Populate properties](#)

Control type: String

Location (x,y): 0 0

Horiz. Anchor: Left

Width: 100

Prompt settings:

Caption: Case Id.

Position: Left

Value settings:

Position: Right

Data settings:

Update location: Veh1

Field name: CaseID

Field type: String

Default:

Expression: CaseID.value

☐ Read only

The Veh1 record also needs to know which fields make it unique in the its (Vehicles) table. In this case, a field called CaseIDVeh1 holds the CaseID field for the Veh1 record. Note that the CaseIDVeh1 field is not visible. Also, the expression is "CaseID.value". This means that when this record is written to the database, the CaseID field will contain the value from the main record's CaseID.. In addition, there is a field called VehNum1 that holds the vehicle number. Similar to the CaseIDVeh1 entry field, this one also has an expression. However in this case, that expression is simply "1". Whenever a Veh1 record is written, it will have a CaseID that matches the Main record, and a VehNum that is 1.

Data settings:

Update location: Veh1

Field name: VehNum

Field type: Integer

Default:

Expression: 1

☐ Read only

- Vehicle2

Update locations:

- Main
- Veh1
- Veh2
- Veh3

Update location name: Veh2

Connection: DataEntry Table name: Vehicles

Do write: `DataCount(Veh2Type.Value, Veh2Dir.Value, Veh2Mov.Value, Veh2SpLmt.Value, Veh2EstSp.Value, Drvr2Act.Value, Drvr2ContrFact.Value, Ped2Act.Value, Veh2Defect.Value, Unit2SafetyEq.Value, Unit2Age.Value) > 0`

The Veh2 update location is very much like Veh1, but it checks for the fields that indicate properties of vehicle 2 instead of vehicle 1.

The unique fields for Veh2 are exactly the same as for Veh1, but the expression for the VehNum is a literal "2". Also note that since all entry fields in a data entry definition must have unique names, the names for these unique fields have been created using the field name and the update location name.

☐ ENTRY ☒ UNIQUE FIELDS for [Veh2]

sample entry field layout: Prompt: Entry Value

ENTRY FIELDS: + - ↑ ↓

CaseIDVeh2
VehNum2

Entry field name: VehNum2 ☐ Visible

Entry settings: [Populate properties](#)

Control type: Unknown

Location (x,y): 0 0

Horiz. Anchor: Left

Width: 100

Prompt settings: Caption: VehNum2 Position: Left

Value settings: Position: Right

Data settings: ☐ Read only

Update location: Veh2 Expression: 2

Field name: VehNum

Field type: Integer

Default:

- Veh3

Update locations:

Main
Veh1
Veh2
Veh3

Update location name: Veh3

Connection: DataEntry Table name: Vehicles

Do write: `DataCount(Veh3Dir.Value, Veh3Mov.Value)`

Veh3 is a bit different in this example, because in this configuration, there are only two fields that relate to vehicle 3. We are only storing the direction and the movement of a third vehicle. Even though only these two fields need to be checked for Do write, Veh3 still has the same two unique fields, similar to Veh1 and Veh2. Again, the only difference is that VehNum is hard-coded to a "3".

When the "Entry" checkbox is checked, the bottom part of the editor is used to add or remove entry fields or set their properties.

There are a number of field and data entry control types.

Typically, the CaseId field, entered on the "New Record" form is presented as a label on the main entry form.

Several field types utilize special control types. They are:

- Date
- Time
- Primary street
- Cross street
- Node (if used)

In addition, there are several control types that are commonly used throughout the form:

- Lookup
- String
- Whole number

For each entry field, select a control type and fill in any other desired fields. All entry fields must have data settings including Update location and field name. Rarely provide a default or expression.

Specific field types

- RecModifiedBy = String, NOT Visible, Default = UserLogin, Expression=UserLogin, NOT read only
- RecModifiedByLbl = Label, Visible, Read Only, Default = UserLogin, No Expression
- DateRecModified = Date, Not Visible, Not read only, Default = NOW, Expression = NOW

- DateRecModifiedLbl = Label, Visible, Read Only, Default = Now, No Expression
- FormCrashDate - Visible, Control=Date, set to Read Only, no expression, default is now
- CrashDate - Not Visible, Not Read Only, Expression is BuildDate(FormCrashDate.Value,FormCrashTime.Value)
- Same with Time fields, except Control=Time
- Latitude - field needs to be wrapped in Number(Latitude.value). This goes for any floating fields (decimals).
- Longitude - same as Latitude - but also MIN will be negative in the U.S.
- Street1 = Visible, Indexed Primary Street - properties = "IntersectionNormalizer=Default"
- Street2 = Visible, Indexed Cross Street - properties = "IntersectionNormalizer=Default" and next line is "RelatedPrimaryStreet=PrimaryStreet" (without quotes and name must be the control name in DE, so could be Street1.

General field settings

- The Unique fields prompts for a case number (only make main one visible, otherwise will get multiples)
- DO NOT MAKE UNIT NUMBERS FOR UNIQUE FIELDS READ ONLY (make them not visible). If you do, it won't read the values from the database.
- Default area - leave blank - unless they are label values being shown like case id , rec modified, etc. (caseid.value)
- Must have a DoWrite that is a read only field that you can check.
- Unique field - ReportNumber = QueryPrefix=DECrashes (name of query for DE to show crashes), also need DECrashes General Query, and DECrashes Field list.
- If doing a calculated field in the study query (ie for getting most severe injury). Don't put that in the data entry (the main crash severity field). Only have the individual ones in there.
- Do not have any "Unknown" control types. Floating fields should have "Decimal" as control type.
- Do not put any spaces in the Entry field name (will cause error).
- Lookups setting for text is LookupSort=StringValue
- Lookups setting for number is LookupSort=NumberValue
- Value Width (prompt settings) - is how much room a lookup value can take next to the entry - generally keep 100.
- This value can be set to 1 for non-lookup fields (including memo fields)
- Width (entry settings) is how much room you have to enter a value. If it's a lookup - generally keep to 30. Dates 70, Streets 200.

Streets Data Entry Def

- RecModifiedBy and DateRecModified - Not Visible
- Must have "Main" as the top reference.

In the data entry program it is beneficial to have some reasonable min/max values for lat/long fields. This table provides min/max values for all US states.

FIP	ST	State	XMin (Longitude)	YMin (Latitude)	XMax (Longitude)	YMax (Latitude)
		Entire US	-171.791111	18.91619	-66.96466	71.357764
01	AL	Alabama	-88.473227	30.223334	-84.88908	35.008028
02	AK	Alaska	-179.148909	51.214183	179.77847	71.365162
60	AS	American Samoa	-171.089874	-14.548699	-168.1433	-11.046934
04	AZ	Arizona	-114.81651	31.332177	-109.045223	37.00426
05	AR	Arkansas	-94.617919	33.004106	-89.644395	36.4996
06	CA	California	-124.409591	32.534156	-114.131211	42.009518
08	CO	Colorado	-109.060253	36.992426	-102.041524	41.003444
69	MP	Northern Mariana Islands	144.886331	14.110472	146.064818	20.553802
09	CT	Connecticut	-73.727775	40.980144	-71.786994	42.050587
10	DE	Delaware	-75.788658	38.451013	-75.048939	39.839007
11	DC	District of Columbia	-77.119759	38.791645	-76.909395	38.99511
12	FL	Florida	-87.634938	24.523096	-80.031362	31.000888
13	GA	Georgia	-85.605165	30.357851	-80.839729	35.000659
66	GU	Guam	144.618068	13.234189	144.956712	13.654383
15	HI	Hawaii	-178.334698	18.910361	-154.806773	28.402123
16	ID	Idaho	-117.243027	41.988057	-111.043564	49.001146
17	IL	Illinois	-91.513079	36.970298	-87.494756	42.508481
18	IN	Indiana	-88.09776	37.771742	-84.784579	41.760592
19	IA	Iowa	-96.639704	40.375501	-90.140061	43.501196
20	KS	Kansas	-102.051744	36.993016	-94.588413	40.003162
21	KY	Kentucky	-89.571509	36.497129	-81.964971	39.147458
22	LA	Louisiana	-94.043147	28.928609	-88.817017	33.019457
23	ME	Maine	-71.083924	42.977764	-66.949895	47.459686
24	MD	Maryland	-79.487651	37.911717	-75.048939	39.723043
25	MA	Massachusetts	-73.508142	41.237964	-69.928393	42.886589
26	MI	Michigan	-90.418136	41.696118	-82.413474	48.2388

27	MN	Minnesota	-97.239209	43.499356	-89.491739	49.384358
28	MS	Mississippi	-91.655009	30.173943	-88.097888	34.996052
29	MO	Missouri	-95.774704	35.995683	-89.098843	40.61364
30	MT	Montana	-116.050003	44.358221	-104.039138	49.00139
31	NE	Nebraska	-104.053514	39.999998	-95.30829	43.001708
32	NV	Nevada	-120.005746	35.001857	-114.039648	42.002207
33	NH	New Hampshire	-72.557247	42.69699	-70.610621	45.305476
34	NJ	New Jersey	-75.559614	38.928519	-73.893979	41.357423
35	NM	New Mexico	-109.050173	31.332301	-103.001964	37.000232
36	NY	New York	-79.762152	40.496103	-71.856214	45.01585
37	NC	North Carolina	-84.321869	33.842316	-75.460621	36.588117
38	ND	North Dakota	-104.0489	45.935054	-96.554507	49.000574
39	OH	Ohio	-84.820159	38.403202	-80.518693	41.977523
40	OK	Oklahoma	-103.002565	33.615833	-94.430662	37.002206
41	OR	Oregon	-124.566244	41.991794	-116.463504	46.292035
42	PA	Pennsylvania	-80.519891	39.7198	-74.689516	42.26986
72	PR	Puerto Rico	-67.945404	17.88328	-65.220703	18.515683
44	RI	Rhode Island	-71.862772	41.146339	-71.12057	42.018798
45	SC	South Carolina	-83.35391	32.0346	-78.54203	35.215402
46	SD	South Dakota	-104.057698	42.479635	-96.436589	45.94545
47	TN	Tennessee	-90.310298	34.982972	-81.6469	36.678118
48	TX	Texas	-106.645646	25.837377	-93.508292	36.500704
78	VI	United States Virgin Islands	-65.085452	17.673976	-64.564907	18.412655
49	UT	Utah	-114.052962	36.997968	-109.041058	42.001567
50	VT	Vermont	-73.43774	42.726853	-71.464555	45.016659
51	VA	Virginia	-83.675395	36.540738	-75.242266	39.466012
53	WA	Washington	-124.763068	45.543541	-116.915989	49.002494
54	WV	West Virginia	-82.644739	37.201483	-77.719519	40.638801
55	WI	Wisconsin	-92.888114	42.491983	-86.805415	47.080621
56	WY	Wyoming	-111.056888	40.994746	-104.05216	45.005904

There are a number of data entry control types available for receiving user input. Each control type supports some or all of the entry properties available. This section describes in detail how each property is used for each control type.

Memo is a multi-line and/or large string

WholeNumber is a numeric data type stored as an integer. It may be positive or negative, but may not contain a decimal or fractional part of a whole number.

A DecimalNumber is a numeric value that may contain a fractional part of a number. This data type is defined internally as a "double" or "floating point" number.

A Date field stores a date + time value. When used in data entry, it displays a date selection box and does not populate the time portion of the datetime field.

The Time entry type is stored as a date + time value, but the entry only accepts the time portion of the value. The date portion remains unpopulated.

Lookup entry fields store an abbreviated value, often a number or single character, but are backed by a lookup table that assigns English text to each value. A list of the English values is presented during entry to assist the user in selecting the appropriate value. Only values that match a value in the lookup table are accepted.

This special data type is used for selecting a street name from the existing streets table. All streets are available for selection. During entry, street names containing match the entered text are displayed for selection. Only existing values may be selected. New streets may not be added into this field.

This entry type is used to enter a cross street. It is special because it is tied to an IndexedPrimaryStreet in order to know which street to look for cross streets on. In all other ways, it acts like an IndexedPrimaryStreet type.

This entry type accepts an alpha-numeric abbreviation for a location name. Node values are frequently numeric and refer to a descriptive location name for an intersection or

landmark. Entry into this field provides a search to locate node names that contain the search text. Internally, only the node id is stored.

A large lookup works just like a lookup except that it specifies its own query name. This makes it more suitable for lookups containing a large number of potential values such as nodes, cities, citations, etc.

The Next ID entry type is not used

The YesNo entry type presents as a lookup with the possible values of "Yes" and "No". Internally this will return an integer value of 1 for "yes" and a 0 for "no". This entry type is commonly used for integer-based boolean fields.

The label entry type displays the value from a field as a read-only label on the form. This type does not accept input and is often used to display caseid or entry date or user name.

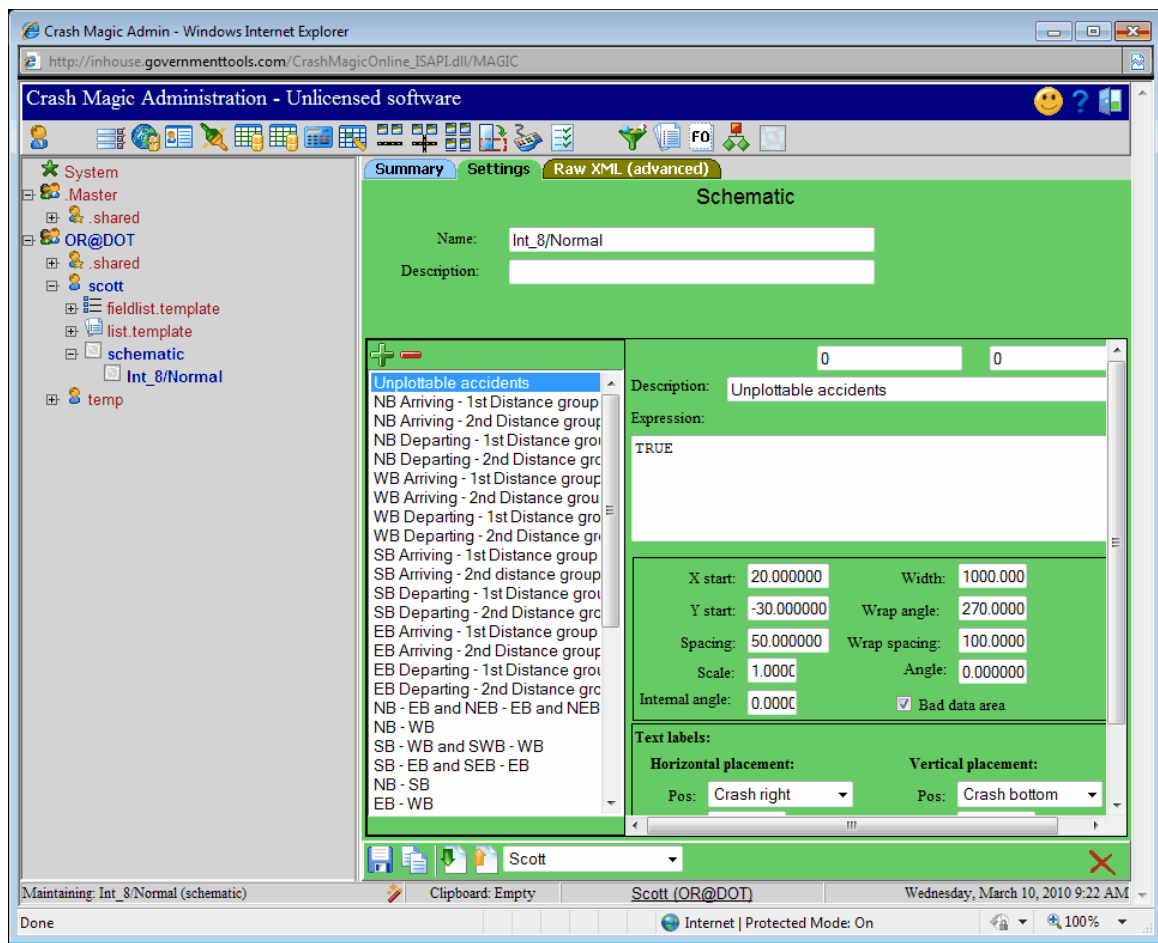
The image type provides the ability to place a raster image on the entry form. This can be used for an agency logo or graphic legend or other situation where an image is needed to aid in entry. This is a static image that references a Crash Magic rasterimage resource.

An annotation is used to display a static piece of text as specified in the data entry definition. This is a read-only piece of text like the label type, but does not reference any field. This entry type is frequently used for headers like "Vehicle 1 Information" or similar.

This is the most basic entry type

Diagram schematic (schematic)

Schematics are used by Crash Magic to define the areas that a diagram uses. Each entry represents a condition. When this condition is met, the symbol (defined in the object map) will be drawn in the current area.



Below the name and description fields of the schematic is an area list and area description section. Above the area list are + and - buttons to add and remove areas. The area description section will display information based on the area selected in the area list. Areas are displayed on a schematic as an isosceles triangle. Collisions are placed at the base of the triangle continuing out to the point.

Description: Name of the area selected. This description will also appears in the area when the show areas check box is checked in the diagram settings tab of a collision diagram.

The screenshot shows the 'Unplottable accidents' configuration window. On the left is a scrollable list of accident types, with 'Unplottable accidents' selected. The main panel has a green background and contains the following fields:

- Description:** Unplottable accidents
- Expression:** TRUE
- X start:** 20.000000
- Width:** 1000.000
- Y start:** -30.000000
- Wrap angle:** 270.0000
- Spacing:** 50.000000
- Wrap spacing:** 100.0000
- Scale:** 1.0000
- Angle:** 0.000000
- Internal angle:** 0.0000
- ☒ **Bad data area**
- Text labels:**
 - Horizontal placement:**
 - Pos:** Crash right
 - Offset:** 8.000000
 - Align:** Center
 - Vertical placement:**
 - Pos:** Crash bottom
 - Offset:** 0.000000
 - Align:** Middle
 - Angle:** 270.00000

Expression: This expression is used to determine if a collision diagram will be placed within the area. Expressions must resolve to a True or False value. A true value will result in the collision being placed in the area. Expressions can also use [calculated fields](#) ²¹⁵.

Unplottable Accidents:

Each collision diagram starts with an Unplottable accidents area. The expression for this area is always true. Collisions that can not be placed in an other area will be placed in the unplottable area.

The following area controls the properties and placement of a collision area:

X start: This is the starting X coordinate point for the collision area.

Y start: This is the starting Y coordinate point for the collision area.

Spacing: This is the spacing between collisions. The default is 50.

Scale: This is the scale of the collision graphics placed in the area.

Internal Angle: The rotation of graphics within an area.

Width: This is width of the collision area. This dictates how large a collision area is on a diagram. A larger area will allow more collisions before wrapping occurs.

Wrap angle: Collisions that do not fit in a single collision area will wrap to a new area. The angle specified in this field specifies where the wrap area will begin.

Wrap spacing: This sets the space for an area if there are more collisions than the width of the area can hold

Angle: This is the angle of the selected collision area. After the first collision is placed additional collisions will be placed at the angle specified.

Bad data area: This check box is to indicate that the area is for unplotable collisions

The following section is for control of collision labels:

Text Labels Horizontal placement

Pos: The horizontal placement of the label in relation to the collision graphic

Offset: This control will move the calculated label placement horizontally

Align: The horizontal alignment of a collision label in relation to the collision graphic

Text Labels Vertical placement

Pos: The vertical placement of the label in relation to the collision graphic

Offset: This control will move the calculated label placement vertically

Align: The vertical alignment of a collision label in relation to the collision graphic

Angle: The angle section in the text label specifies that angle of a collision label.

Curb lines:

Curb lines are specified in the Raw XML in an SVG element tag. In the following example creates curb lines for a four way roundabout intersection.

```
<svg xmlns="http://www.w3.org/2000/svg">
  <g>
    <path d="M 0,200 L 150,200 A50,50 0 0,1 200,150 L 200,0
    <path d="M 0,200 L 150,200 A50,50 0 0,1 200,150 L 200,0
    <circle cx="500" cy="500" r="200" style="fill:white;stroke:black;"/>
    <path d="M 0,498 L 190,498
    <g transform="translate( 825, 825 )">
      <text id="CorridorLabel" transform="scale(1,-1) rotate(-45)"/>
    </g>
  </g>
</svg>
```

Help reference (helpreference)

The Help Reference data provides links between the program and the help document.

```
<pdroot>
  <HelpReferenceTable>
    <General>
      <BaseURL>/files/Help/HTML</BaseURL>
    </General>
    <HelpItems>
      <!-- ***** Analysis side help keywords ***** -->
      <HelpItem Keyword="cmMainForm">
        <Title>Crash Magic Online</Title>
        <Description>Basic information about Crash Magic</Description>
        <URL>IntroductionOverview.htm</URL>
        <OtherKeywords>
          <Keyword>CurrentWebManual</Keyword>
          <Keyword>PDFManual</Keyword>
          <Keyword>Tutorial</Keyword>
          <Keyword>PSROverview</Keyword>
        </OtherKeywords>
      </HelpItem>
      <HelpItem Keyword="cmDefaultPanel">
        <Title>Home Page</Title>
        <Description>Home page</Description>
        <URL>Homepagepanel.htm</URL>
      </HelpItem>
      <HelpItem Keyword="cmProjectPanel">
        <Title>Project panel</Title>
        <Description>Project panel overview</Description>
        <URL>PSRProjectpanel.htm</URL>
        <OtherKeywords>
          <Keyword>Panelbuttons</Keyword>
        </OtherKeywords>
      </HelpItem>
      <HelpItem Keyword="cmDiagramPanel">
        <Title>Diagram panel overview</Title>
        <Description>Basic diagram information</Description>
        <URL>DiagramPanelOverview.htm</URL>
        <OtherKeywords>
          <Keyword>Filter</Keyword>
          <Keyword>Templates</Keyword>
          <Keyword>Panelbuttons</Keyword>
        </OtherKeywords>
      </HelpItem>
      <!-- ***** Admin side help keywords ***** -->
      <HelpItem Keyword="cmMainAdminForm">
        <Title>Administration form</Title>
        <Description>Overview of administrative functions</Description>
        <URL>AdministrationOverview.htm</URL>
      </HelpItem>
      <HelpItem Keyword="cmAdminRolePanel">
        <Title>Database roles</Title>
        <Description>Help with database role objects</Description>
        <URL>PSRAttr_Role.htm</URL>
      </HelpItem>
      <HelpItem Keyword="cmAdminSchematicPanel">
        <Title>Diagram schematics</Title>
```



```

        <Description>Help with diagram schematics</Description>
        <URL>PSRAttr_Schematic.htm</URL>
    </HelpItem>
    <!-- ***** User_defined help keywords ***** -->
    <HelpItem Keyword="UserDefined">
        <Title>User-specific help topic</Title>
        <Description>Base level user defined help link</Description>
        <URL>help_system_description.htm</URL>
    </HelpItem>
</HelpItems>
</HelpReferenceTable>
<Settings/>
</pdroot>

```

Import - DB to XML (dbtoxml)

Allows users to import files into an XML structure from a data source for future processing by the XML to DB process.

Import - DB to XML PSRattrs can be created using the [Configuration Helper](#)²²⁴

Current XML definition for XML files that can be imported by Crash Magic. The purpose of this section is to document the XML structures used by the converter. Clients should not alter the XML directly. The XML defines how the data is queried and how the generated XML file is defined. All XML files generated by the Crash Magic Converter will begin with "<pdroot>" as the root element.

```

<DbToXml>
  <General>
    <FileRenames>
      <FileRename>*Incident*=Incident.txt</FileRename>
      <FileRename>*Unit*=Unit.txt</FileRename>
      <FileRename>*Person*=Person.txt</FileRename>
    </FileRenames>
    <FileChecks>
      <FileCheck>
        <FileName>Incident.txt</FileName>
        <FirstLine>IncidentID, Microfilm, ADOTReceivedDate, Status, DataCon
      </FileCheck>
      <FileCheck>
        <FileName>Unit.txt</FileName>
        <FirstLine>IncidentID, UnitID, UnitNumber, UnitType, UnitTypeDesc, T
      </FileCheck>
      <FileCheck>
        <FileName>Person.txt</FileName>
        <FirstLine>IncidentID, UnitID, UnitNumber, PersonID, PersonNumber, P
      </FileCheck>
    </FileChecks>
  </General>

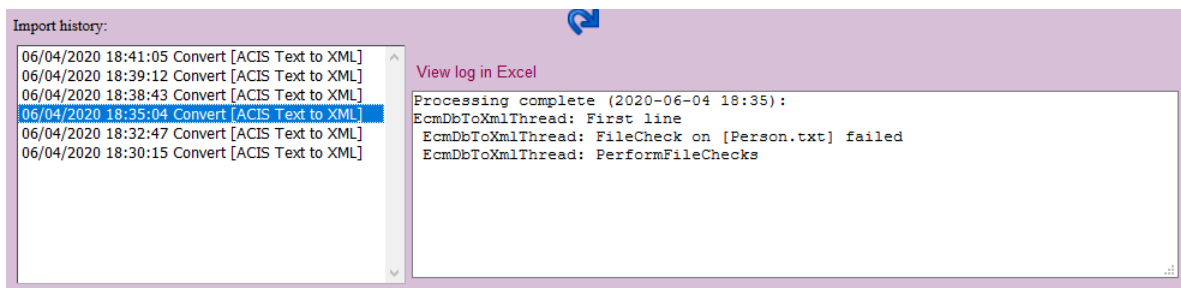
```

Example start of the XML file

- **FileRenames** - (Optional) The connection string(s) in the TableSpecs below this section must specify exact file name(s). For some clients, the filenames may be different / unique in each extract. The FileRenames section provides the ability to

rename the files found using wildcards to expected file names. If the file names in the zip file are always the same, then this section is not necessary and may be left blank. *It is critical that there is not more than one file that will be renamed in each case. This would not be supported by the importer anyway, but will cause problems even in the FileRenames section.*

- FileRename contains <FileSpec>=<StaticFileName> For example Incident*=Incident.txt will rename any files that start with "Incident" to the name "Incident.txt". In the example above, as long as the word Incident appears anywhere in the file name, it will be renamed to Incident.txt.
- **FileChecks** - (Optional, but recommended) Each FileCheck applies to one of the files in the import. The properties in this section can be used to validate the file prior to attempting the convert to XML process. Any failures here will raise an error before the convert process starts
 - **FileName** - This is the name of the file after the FileRenames occur. So this is the name of the file that will be used in the TableSpecs below. It should be a simple file name without a path.
 - **FirstLine** - In typical text/CSV files that are used for data transfer, the first line of the file is called a "header" line and contains the names of the field that will be found on following lines. This sequence is critical to Crash Magic's converter. The FileChecks section can be used to validate this header line and make sure that if the field order or fields themselves have not changed.
 - This entry must match the first line of the file being imported exactly.
 - It is case-sensitive and even extra/missing spaces within the line will cause it to fail.
 - The "..." in the example above is just for display purposes in the manual
 - A blank entry will always cause an error to be raised
 - To include an entry without a FirstLine check, the single character asterisk * may be used. This will match any content in the line and will not generate an error. This practice is not recommended if it can be avoided.



This is an example of the error that will be shown upon a FileCheck error

	A	B	C	D	E
1	MacroArea	LogDate	LogLevel	MicroArea	Message
2	Import	6/4/2020 18:41	Important	Convert [ACIS Text to XML]	BEGIN ConvertThread
3	Import	6/4/2020 18:41	Important	ID	Convert [ACIS Text to XML]
4	Import	6/4/2020 18:41	Important	Convert [ACIS Text to XML]	BEGIN UnZip
5	Import	6/4/2020 18:41	Important	Convert [ACIS Text to XML]	Unzipped 3 files from Mesa01012019.hzip
6	Import	6/4/2020 18:41	Important	Convert [ACIS Text to XML]	END UnZip
7	Import	6/4/2020 18:41	Important	Convert [ACIS Text to XML]	BEGIN FileRenames
8	Import	6/4/2020 18:41	Info	ApplyRenames	Renaming "Incident*" to "Incident.txt"
9	Import	6/4/2020 18:41	Info	ApplyRenames	Renaming "Unit*" to "Unit.txt"
10	Import	6/4/2020 18:41	Info	ApplyRenames	Renaming "Person*" to "Person.txt"
11	Import	6/4/2020 18:41	Important	Convert [ACIS Text to XML]	END FileRenames
12	Import	6/4/2020 18:41	Important	Convert [ACIS Text to XML]	BEGIN RemoveBOMs
13	Import	6/4/2020 18:41	Important	Convert [ACIS Text to XML]	END RemoveBOMs
14	Import	6/4/2020 18:41	Important	Convert [ACIS Text to XML]	BEGIN PerformFileChecks
15	Import	6/4/2020 18:41	Info	FileCheck	Checking "Incident.txt"
16	Import	6/4/2020 18:41	Info	FileCheck	Checking "Unit.txt"
17	Import	6/4/2020 18:41	Info	FileCheck	Checking "Person.txt"
18	Import	6/4/2020 18:41	Important	Convert [ACIS Text to XML]	END PerformFileChecks
19	Import	6/4/2020 18:41	Important	Convert [ACIS Text to XML]	BEGIN WriteRequiredFiles
20	Import	6/4/2020 18:41	Important	Convert [ACIS Text to XML]	END WriteRequiredFiles
21	Import	6/4/2020 18:41	Important	Convert [ACIS Text to XML]	BEGIN ConvertProcess
22	Import	6/4/2020 18:41	Debug		Creating output folder

This is an example of a successful start to the import process. Note each of the FileCheck entries indicating each of the files that were checked.

Defining XML structure of the the file to be created.

```

<TableSpecs>
  <TableSpec>
    <Name>Accidents</Name>
    <Primary>True</Primary>
    <ConnectionString>Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\ImportData;
Extended Properties="text;HDR=NO;FMT=Delimited"</ConnectionString>
    <Login/>
    <Password/>
    <RequiredFiles>
      <RequiredFile>Import schema.ini</RequiredFile>
    </RequiredFiles>
    <BaseName>CrashRecord</BaseName>
    <GroupName/>
    <ItemName>Environment</ItemName>
    <UniqueFields>
      <UniqueField>IncidentID</UniqueField>
    </UniqueFields>
    <ChildNames>
      <ChildName>Vehicles</ChildName>
    </ChildNames>
    <QueryLines>
      <QueryLine>SELECT</QueryLine>
      <QueryLine>IncidentID,</QueryLine>
      <QueryLine>IncidentDate,</QueryLine>
      <QueryLine>IncidentDateTime,</QueryLine>
      <QueryLine>CollisionManner,</QueryLine>
      <QueryLine>LightCondition,</QueryLine>
      <QueryLine>FirstHarmfulEvent,</QueryLine>
      <QueryLine>FirstHarmfulLocation,</QueryLine>
      <QueryLine>TotalUnits,</QueryLine>
      <QueryLine>TotalInjuries,</QueryLine>
      <QueryLine>TotalFatalities,</QueryLine>
      <QueryLine>InjurySeverity,</QueryLine>
      <QueryLine>AlcoholInvolvementFlag,</QueryLine>
      <QueryLine>DrugInvolvementFlag,</QueryLine>
      <QueryLine>HazardousFlag,</QueryLine>
      <QueryLine>HitAndRunFlag,</QueryLine>

```

```

<QueryLine>Onroad,</QueryLine>
<QueryLine>CrossingFeature,</QueryLine>
<QueryLine>MPNum,</QueryLine>
<QueryLine>CityId,</QueryLine>
<QueryLine>CountyId,</QueryLine>
<QueryLine>StateId,</QueryLine>
<QueryLine>StateCode,</QueryLine>
<QueryLine>CountryCode,</QueryLine>
<QueryLine>Latitude,</QueryLine>
<QueryLine>Longitude,</QueryLine>
<QueryLine>IntersectionType,</QueryLine>
<QueryLine>JunctionRelation,</QueryLine>
<QueryLine>Weather</QueryLine>
<QueryLine>FROM CrashEnvironment#txt</QueryLine>
<QueryLine>WHERE IncidentDate BETWEEN '1/1/2012' AND '1/15/2012'</QueryLine>
<QueryLine>ORDER BY IncidentID</QueryLine>
</QueryLines>
</TableSpec>

```

Example specification of the primary XML element

TableSpec - Specifies the start of the xml structure that will be created.

Name - Name of the of the table specification

Primary - Boolean value that indicates if this is the primary XML node.

ConnectionString - Defines the connection to the data source.

Login - Login name if required by the data source.

Password - Password for the login if required by the data source.

RequiredFile - Name of and required Utility - Text lines PSRattr for import (In this example a [schema.ini](#) file has been added to a Utility -Text line with the name of "Import schema.ini").

BaseName - Starting node of the XML crash record

GroupName - Used to define a group of XML nodes for example a value of Vehicles can be used to define a group of Vehicle children records

ItemName - The name of the XML node that will contain elements from the select statement.

QueryLine - Query to extract records from source. Each field returned from the select statement creates an XML element.

UniqueField - The primary identifier for the record. This field will also be used for name of the XML file that is created.

ChildName - The TableSpec name of XML children that will be included in the XML file.

Import - XML to DB (xmldb)

Allows users to import XML data files into the Crash Magic collision database.

Sample XML to DB

```

<pdroot>
  <XmlToDb>
    <General>
      <ConnectionName>Data Entry</ConnectionName>
      <XMLRepository>D:\Temp</XMLRepository>
      <XMLFileMask>*.xml</XMLFileMask>
      <DocStartXP>.</DocStartXP>
      <RecordListStartXP>.</RecordListStartXP>
      <RecordItemName>Crashes</RecordItemName>
      <RecordStartXP>.</RecordStartXP>
      <DuplicateAction>OverwriteExisting</DuplicateAction>
    </General>
    <RowSpecs>
      <RowSpec>
        <Name>Collision</Name>
        <TableName>Collisions</TableName>
        <ListStartXP>.</ListStartXP>
        <ItemName>.</ItemName>
        <UniqueObjectColumnNames>
          <ColumnName>CaseId</ColumnName>
        </UniqueObjectColumnNames>
        <UniqueTableColumnNames>
          <ColumnName>CaseId</ColumnName>
        </UniqueTableColumnNames>
        <AutoRecords>
          <AutoRecord Name="StreetsCollection">
            <TableName>Streets</TableName>
            <Columns>
              <Column Name="StreetNum" IsNextId="True" />
              <Column Name="StreetName" />
            </Columns>
          </AutoRecord>
        </AutoRecords>
        <Columns>
          <Column>
            <ColumnName>CaseId</ColumnName>
            <ValueXPExpression>impRowNodeText( "CaseId" )</ValueXPExpression>
            <LookupMethod>none</LookupMethod>
            <LookupReference/>
          </Column>
          <Column>
            <ColumnName>Street1</ColumnName>
            <LookupMethod>AutoRecord</LookupMethod>
            <LookupReference>StreetsCollection</LookupReference>
            <ValueXPExpression Column="StreetName">impRowNodeText( "StreetName" )</ValueXPExpression>
          </Column>
          <Column>
            <ColumnName>Street2</ColumnName>
            <LookupMethod>AutoRecord</LookupMethod>
            <LookupReference>StreetsCollection</LookupReference>
            <ValueXPExpression Column="StreetName">impRowNodeText( "StreetName" )</ValueXPExpression>
          </Column>
        </Columns>
        <SubSpecNames>
          <SubSpecName>VehicleOne</SubSpecName>
        </SubSpecNames>
      </RowSpec>
    </RowSpecs>
    <SubSpecs>
      <RowSpec>
        <Name>VehicleOne</Name>

```

```

<TableName>Vehicles</TableName>
<DoWriteExpression>true</DoWriteExpression>
<ListStartXP>.</ListStartXP>
<ItemName>.</ItemName>
<UniqueObjectColumnNames>
  <ColumnName>CaseId</ColumnName>
</UniqueObjectColumnNames>
<UniqueTableColumnNames>
  <ColumnName>CaseId</ColumnName>
  <ColumnName>VehNum</ColumnName>
</UniqueTableColumnNames>
<AutoRecords/>
<Columns>
  <Column>
    <ColumnName>CaseId</ColumnName>
    <ValueXPExpression>impRowNodeText( "CaseId" )</ValueXPExpression>
    <LookupMethod>none</LookupMethod>
    <LookupReference/>
  </Column>
  <Column>
    <ColumnName>VehNum</ColumnName>
    <ValueXPExpression>1</ValueXPExpression>
    <LookupMethod>none</LookupMethod>
    <LookupReference/>
  </Column>
  <Column>
    <ColumnName>UnitType</ColumnName>
    <ValueXPExpression>impRowNodeText( "UnitType" )</ValueXPExpression>
    <LookupMethod>none</LookupMethod>
    <LookupReference/>
  </Column>
  <Column>
    <ColumnName>TravelDirection</ColumnName>
    <ValueXPExpression>impRowNodeText( "TravelDirection" )</ValueXPExpression>
    <LookupMethod>none</LookupMethod>
    <LookupReference/>
  </Column>
</Columns>
<SubSpecNames>
  <SubSpecName>PersonRecord</SubSpecName>
</SubSpecNames>
</RowSpec>
<RowSpec>
  <Name>PersonRecord</Name>
  <TableName>Person</TableName>
  <DoWriteExpression>true</DoWriteExpression>
  <ListStartXP>Persons</ListStartXP>
  <ItemName>Person</ItemName>
  <DoWriteExpression>true</DoWriteExpression>
  <ListStartXP>.</ListStartXP>
  <ItemName>.</ItemName>
  <UniqueObjectColumnNames>
    <ColumnName>CaseId</ColumnName>
  </UniqueObjectColumnNames>
  <UniqueTableColumnNames>
    <ColumnName>CaseId</ColumnName>
    <ColumnName>VehNum</ColumnName>
    <ColumnName>PersonNumber</ColumnName>
  </UniqueTableColumnNames>
  <Columns>
    <Column>

```

```

        <ColumnName>CaseId</ColumnName>
        <ValueXPExpression>impRowNodeText( "CaseId" )</ValueXPExp
        <LookupMethod>none</LookupMethod>
        <LookupReference/>
    </Column>
    <Column>
        <ColumnName>VehNum</ColumnName>
        <ValueXPExpression>1</ValueXPExpression>
        <LookupMethod>none</LookupMethod>
        <LookupReference/>
    </Column>
    <Column>
        <ColumnName>PersonID</ColumnName>
        <ValueXPExpression>impRowNodeText( "PersonID" )</ValueXP
        <LookupMethod>none</LookupMethod>
        <LookupReference/>
    </Column>
    <Column>
        <ColumnName>PersonNumber</ColumnName>
        <ValueXPExpression>impRowNodeText( "PersonNumber" )</Valu
        <LookupMethod>none</LookupMethod>
        <LookupReference/>
    </Column>
    <Column>
        <ColumnName>PersonType</ColumnName>
        <ValueXPExpression>impRowNodeText( "PersonType" )</Value2
        <LookupMethod>none</LookupMethod>
        <LookupReference/>
    </Column>
</Columns>
<SubSpecNames/>
</RowSpec>
</SubSpecs>
</XmlToDb>
<Settings/>
</pdroot>

```

General section - The general section is used to define the starting point of the XML document being used for Import.

ConnectionName: Name of the Crash Magic connection that will be used to insert records into the database.

XMLRepository: This feature has been deprecated.

XMLFileMask: Mask to read the files that are to be imported(*.xml will default to reading all .xml files uploaded.)

DocStartXP: The XPath definition of the starting point of the xml document.

RecordListStartXP: The XPath definition for the parent node to a list of child nodes(This is used for clients with multiple records in a single XML file).

RecordItemName: Child node of the node list defined in the RecordListStartXP.

RecordStartXP: XPath definition that defines the starting point for selecting records. This is the default starting point when RecordListStartXP and RecordItemName are left blank.

DuplicateAction: Action to be taken if the record already exists in the database. Options are:

- Skip - Skip importing the record
- OverwriteExisting - Deletes the record from the database and imports the record from the xml file
- UpdateExisting -
- LeaveExisting -

Row specification section - The Row specification section specifies a record to be imported.

RowSpec: Start of a record import.

Name: Name of the RowSpec.

TableName: The name of the table that records will be imported into.

DoWriteExpression: Expression that determines when a record should be written.

ListStartXP: The XPath definition of a parent node to a list of child nodes(In the Unit RowSpec the XML has a Units section with one or more Unit).

ItemName: Child node of the node list defined in ListStartXP.

UniqueObjectColumnNames: List of columns that determine a unique record.

ColumnName: The name of a column that determines a unique record.

UniqueTableColumnNames: List of unique fields of the table being imported into(Primary key fields for the table).

ColumnName: The field name of the table that determines a unique record.

Auto Records specification section - This section is used to maintain a list of items that will be added to. In this example auto records are being used to maintain a list of streets from the streets table. If the street name exists in the data the Id(StreetNum field) will be imported to the record. If the street name does not exist the street will be added to the Streets table, and the Id for the new street will be imported into the Collisions table.

AutoRecords: Specifies the start of an auto record section

AutoRecord: The Name attribute of this element is used to name the auto record section

TableName

Columns

Column

Columns: Specifies the list of fields that data will be added to.

Column: Specifies a column that will be updated.

ColumnName: Name of the field in the table that will have data added inserted.

ValueXPExpression: Expression used to define the data imported into the field.

There are two functions that can be used to retrieve values from the XML:

ImpRowNodeText(): Starts at the current XML position and retrieves a value from the XML node specified in the function.

ImpRecordNodeText(): Starts at the RecordStartXP of an XML record. This import function is used when a SubSpec record requires a value from a parent node.

LookupMethod: Specifies the method that should be used to lookup a value. Possible values are none, , lookup, streetnormalizer, streetnormalizerlu, and nodenormalizer.

none - No lookup in use for the column

lookup - Performs a lookup for the data provided. This causes the lookup to return an Id for a human readable value passed.

<AutoRecord attribute name>

LookupReference: Specifies name of the lookup field or normalizer that will be used.

SubSpecNames: Used to specify child records that must be imported first.

SubSpecName: Name of the child RowSpec that will be imported before the current record import.

Layout helpers (layouthelpers)

Layout Helpers are used in the layout editor to automatically fill in expressions. There is currently no editor for helpers so the XML must be edited directly.

Helper Tags:

Helper: This is the container tag, each helper is wrapped in a Helper tag.

Name: The friendly name of the helper, this is displayed on the layout panel.

Description: A short description of what the helper does. This will also be displayed on the layout panel when a helper is selected.

Help Keyword: The keyword is used to link directly into the help system. ✖

Expression: A valid crash magic expression. %s can be used where Project, Study, and Report need to be inserted by the layout editor.

Providers: Wrapper tag that holds each provider.

Provider: The object type that this helper can be used for.

Categories: Categories are used to group helpers so they can be easily found from the layout panel. ✖

```
<pdroot>
  <Helpers>
    <Helper>
```

```

        <Name>Study Record Count</Name>
        <Description>The number of records in the selected study.</Description>
        <HelpKeyword>Study.RecordCount</HelpKeyword>
        <Expression>Study.RecordCount.ASCII("Project:%s;Study:%s;ObjType:marrs", "")<
        <Providers>
            <Provider>marrs</Provider>
        </Providers>
        <Categories>
            <Category>Counts</Category>
            <Category>Study</Category>
        </Categories>
        <ForUseIn>ASCII</ForUseIn>
    </Helper>
    <Helper>
        <Name>Study Record Count</Name>
        <Description>The number of records in the selected study.</Description>
        <HelpKeyword>Study.RecordCount</HelpKeyword>
        <Expression>Study.RecordCount.ASCII("Project:%s;Study:%s;ObjType:intersection
        <Providers>
            <Provider>intersection</Provider>
        </Providers>
        <Categories>
            <Category>Counts</Category>
            <Category>Study</Category>
        </Categories>
        <ForUseIn>ASCII</ForUseIn>
    </Helper>
    <Helper>
        <Name>Query Name</Name>
        <Description>Name of the query used by the study</Description>
        <HelpKeyword>Study.QueryName.ASCII</HelpKeyword>
        <Expression>Study.QueryName.ASCII("Project:%s;Study:%s;ObjType:daterange", ""
        <Providers>
            <Provider>daterange</Provider>
        </Providers>
        <Categories>
            <Category>Study</Category>
        </Categories>
        <ForUseIn>ASCII</ForUseIn>
    </Helper>
    <Helper>
        <Name>List</Name>
        <Description>Crash Magic List</Description>
        <HelpKeyword>Report.List</HelpKeyword>
        <Expression>ApplyXSL( List.XML("Project:%s;Study:%s;Report:%s;ObjType:list",
        <Providers>
            <Provider>list</Provider>
        </Providers>
        <Categories>
            <Category>List</Category>
        </Categories>
        <ForUseIn>FO</ForUseIn>
    </Helper>
</Helpers>
<Settings/>
</pdroot>

```

Locale info (localeinfo)

Creating a Default Locale Info will allow the location information to be set. This is designed for users outside of the United States. Without a locale object the User Group will default to English United States. Setting the locale information will determine how Crash Magic displays data to users. Setting this information will drive how date, time and month fields are displayed in reports created within Crash Magic. If Locale Info is not created or selected Crash Magic will default to the settings of the server.

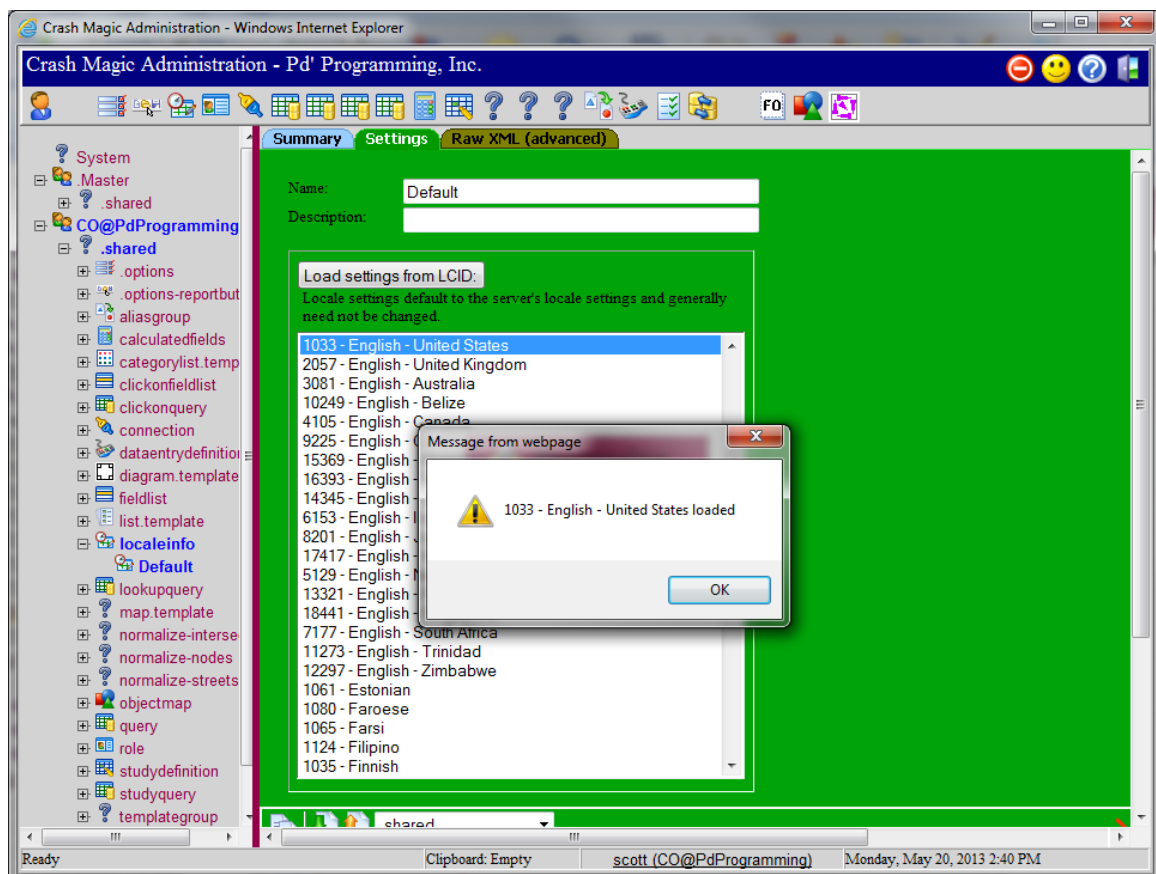
To create a Locale Info ensure the .shared user created is selected in the admin tree, and click on the Locale

Info button .

Name: The name of the Locale Info that will be used. If not already named changed the name to Default.

Load settings from LCID: Will load the location information from the location selected in the list below.

The list displayed on the panel contains location information based on Language and region. Select the language and region that Crash Magic will be used. Then click the Load Settings from LCID button. Click OK when the settings have been loaded.



In this image English - United States has been selected

Normalizer - Intersections (normalize-intersections)

The intersections normalizer in Crash Magic is used by the program to provide a list of standard intersection names. Any time a list of intersections is selected or displayed the intersections normalizer has provided the list of the intersections.

Name: This is the name of the intersection normalizer.(This name should be the same name as the Node normalizer when collision records contain node and intersection location data. Pd Programming recommends the name local for local intersection street names. This will allow users to switch to a Node High Crash Location list.)

Description: This field contains a description of the intersection normalizer.

Normalize Streets: This drop down menu is used to select the name of the streets normalizer that will be used by this intersection normalizer.

Distinct intersections query: This drop down is used to select the [intersection list](#)²⁷⁴ query.

Case Sensitive: Indicates that this normalizer should respect the case of street names.

Primary Street: This drop down is used to select the field used as the primary street from the distinct intersections query

Unique 1 (optional): This drop down allows the selection of a field from the distinct intersection query that specifies the primary street is unique to region like a county.

Unique 2 (optional): This drop down allows the selection of a field from the distinct intersection query that specifies the primary street is unique to an additional region. For example if Unique 1 specifies the county that an intersection is located Unique 2 can specify the intersection is also unique to a city.

Cross Street: This drop down is used to select the field used as the cross street from the distinct intersections query

Unique 1 (optional): This drop down allows the selection of a field from the distinct intersection query that specifies the cross street is unique to region like a county.

Unique 2 (optional): This drop down allows the selection of a field from the distinct intersection query that specifies the cross street is unique to an additional region.

Normalizer - Nodes (normalize-nodes)

The node normalizer in Crash Magic is used by the program to provide a list of standard node names. Any time a list of nodes is selected or displayed the nodes normalizer has provided the list of of the intersections.

Name: This is the name of the node normalizer.(This name should be the same name as the Intersection normalizer when collision records contain node and intersection location data. Pd Programming recommends the name local for local node intersection names.)

Description: This field contains a description of the node normalizer.

Unique Delimiter: This value is used to separate the Node names from the unique 1 and 2 values when displayed to the user.

Unique Starters: This value is used to enclose the unique 1 and 2 values when displayed to the user.

Friendly Name Delimiter: This value is used to separate the node with the friendly name of the node that is displayed to the user.

Distinct Query: This drop down menu is used to specify the query of distinct node locations.

Case Sensitive: Indicates that this normalizer should respect the case of node names

Node: This drop down menu is used to specify the field from the distinct nodes query that contains the node number. The drop down menu to the right specifies the data type of the field.

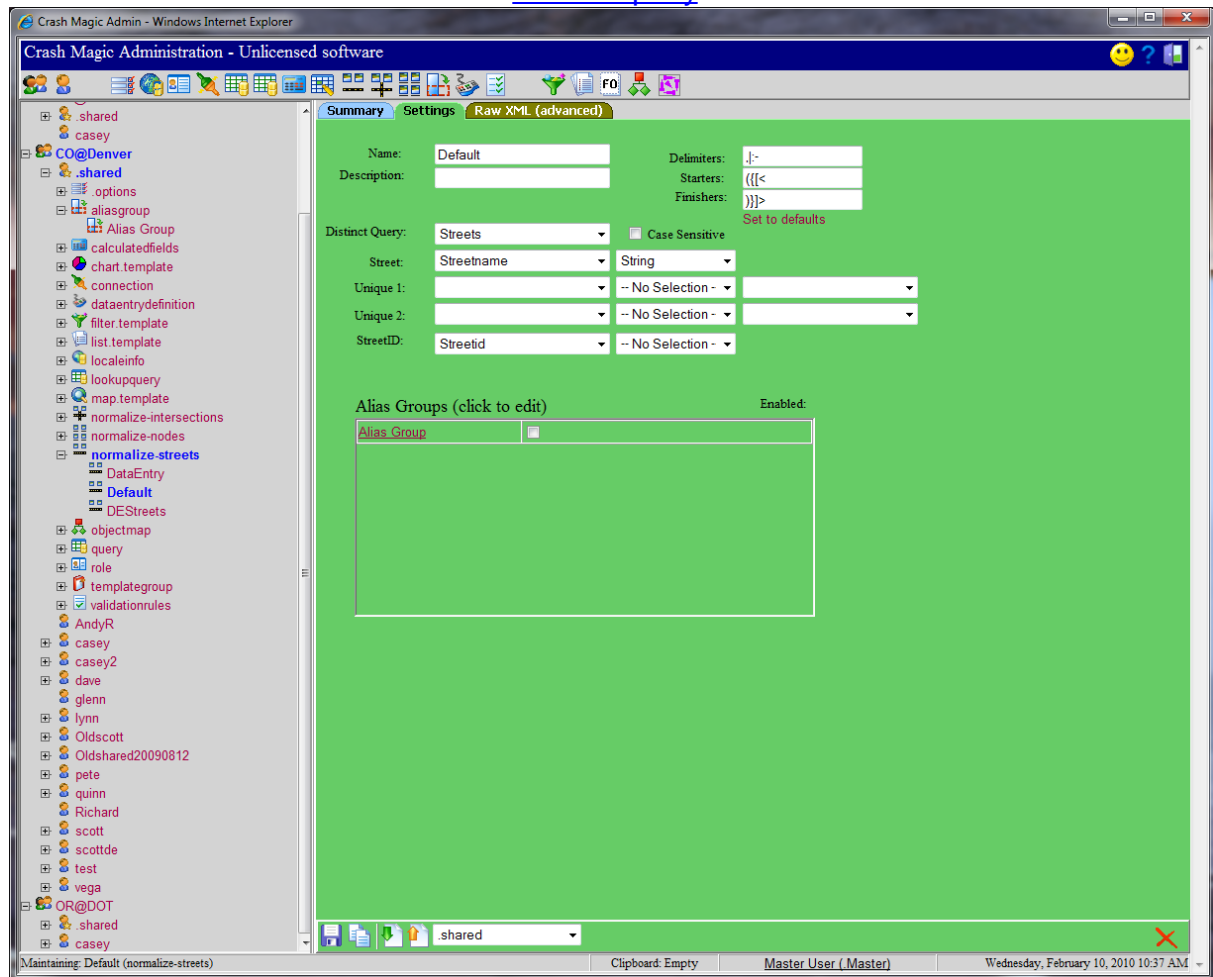
Unique 1: The first drop down allows the selection of a field from the distinct nodes query that specifies the node is unique to region like a county. The drop down menu to the right specifies the data type of the field. The last drop down box allows the user to select a lookup query for the unique 1 field.

Unique 2: This drop down allows the selection of a field from the distinct node query that specifies the node is unique to an additional region. For example if Unique 1 specifies the county that an node is located Unique 2 can specify the node is also unique to a city. The drop down menu to the right specifies the data type of the field. The last drop down box allows the user to select a dependent lookup query for the unique 2 field.

Friendly name: This drop down allows that selection of a friendly name field from the distinct nodes query. This friendly name will also be displayed to the user.

Normalizer - Streets (normalize-streets)

The streets normalizer in Crash Magic is used by the program to provide a list of standard street names. Any time a street is selected in a study the streets normalizer has provided the list of streets for the user to select. The streets normalizer defines which fields should be used from the [Streets query](#)²⁷⁴ to define a standard list of streets.



Name: This is the name of the streets normalizer.

Description: This field contains a description of the streets normalizer.

Unique Delimiter: This value is used to separate the Street names from the unique 1 and 2 values when displayed to the user.

Unique Starters: This value is used to enclose the unique 1 and 2 values when displayed to the user.

Distinct Query: Query that will return an entire list of streets

Case Sensitive: Indicates that this normalizer should respect the case of street names

Street: Field from the Distinct query that returns the name of the street

Unique1: Field from the streets query that indicates the street is unique to an area. For example a street may cross several counties. This field could be used to specify the county. This field should be left blank if this information does not exist in the collision data. Along with specify the field from the distinct query the type of field needs to be selected. A drop down menu of [lookup queries](#)^[275] is also provided. The drop down menu on the far left allows the user to select a lookup for this field. This means that if the field from the streets query contains a number that represents the location. The lookup query selected in this drop down menu will be used to display the readable value to the user. Study queries that use Unique1 must have [PrimaryStreetUnique1](#)^[277], [CrossStreetUnique1](#)^[278] and/or [RouteUnique1](#)^[277] fields flagged in the study query.

Unique2: Field from the Distinct query that indicates the street is unique to an area within the Unique 1 area. For example a street may cross cities within a county. This field could be used to specify the city where the street is located. This field should be left blank Unique 1 is not in use. The middle drop down box on this page also allows the data type for the field to be specified. The drop down menu on the left allows a dependent query to be specified. The [dependent lookup query](#)^[276] allows Crash Magic to retrieve a street using the street name and the value in Unique1. Study queries that use Unique1 must have [PrimaryStreetUnique2](#)^[278], [CrossStreetUnique2](#)^[278] and/or [RouteUnique2](#)^[277] fields flagged in the study query.

Note: Unique 1 and Unique 2 are not required fields but are often needed if the database contains more than one municipality. The Unique 2 field relies on the Unique 1 field, and is always a subset of the Unique 1 field. The most common use for these fields would be Unique 1 as "county" and Unique 2 as "city".


StreetID: ID for the street selected. This field should be left blank if street names do not have a corresponding value to identify the street.

Alias Groups: Allows clients to define street alias names for a location. As an example HWY 93 could cross a town where it is call Main St. Alias groups enable the ability to define a street using a single street name. An [Alias Group](#)^[214] must be created to store the alias information before this feature can be used.

Enable: This enables the Alias Group selected to be used by the normalizer.



Visible: This makes the Alias Group streets visible to users in the street selection boxes.

Object map (objectmap)

The object map  is used to map graphic objects to a collision diagram.

Name: Name of the object map. We recommend default.

Description: Description of the object map.

In the object map settings tab use the  to create a new objects for display on collision diagrams. Use the minus button  to remove selected graphic objects from the list.

Description: Name shown in list of objects. This should describe what the object is showing Example Graphic for Animal.

Expression: This expression determines when the object should be displayed. This expression must be a Boolean. This means that it must resolve to true or false.

Example:

```
AnyMatch( VehManeuverCode_One, 5 )
```

The previous example checks the field VehManeuverCode_One in a collision record. If the field contains a 5 the symbol is drawn for the collision record. Any value other than a five will not cause the symbol to be drawn.

Insert: This button inserts the symbol selected from the drop down list into the symbol window.

Symbol: This area shows the symbol to be displayed. This can also be an expression Example:

```
If(Veh1IsBike,  
_symBikeLeft,  
_symLeftTurn)
```

The previous expression uses the calculated field Veh1IsBike to display a bike turning left symbol. Otherwise the standard left turn vehicle symbol is displayed.

Is Vehicle: This determines if an object should be rotated and moved as a vehicle. Selecting a value of one informs Crash Magic that the object to be displayed is the first vehicle in the collision.

With Vehicle: This determines if an object is part of a vehicle and should be moved with the vehicle.

Angle: If a a symbol is not a vehicle or with a vehicle the object is rotated based on this field.

Color: Not currently used.

Thickness: This sets the thickness of the object lines being drawn for the object.

Opacity: Sets the opacity of the object being drawn.

Default symbol map for objects based on MMUCC First Harmful Event

Event	Symbol
Non-Collisions	
Overturn/Rollover	
Fire/Explosion	
Immersion	
Jackknife	
Cargo/Equipment Loss or Shift	
Fell/Jumped from Motor Vehicle	
Thrown or Falling Object	
Other Non-Collision	
Collision with Person, Motor Vehicle, or Non-Fixed Object:	
Pedestrian	Pedestrian
Pedalcycle	Bicycle
Railway Vehicle (train, engine)	General "Fixed Object" (if not coded as a vehicle)
Animal	Animal
Motor Vehicle in Transport	Vehicle (based on vehicle movement)
Parked Motor Vehicle	Parked Vehicle
Work Zone/Maintenance Equipment	General "Fixed Object"
Other Non-Fixed Object	General "Fixed Object"
Collision with Fixed Object:	
Impact Attenuator/Crash Cushion	General "Fixed Object"
Bridge Overhead Structure	General "Fixed Object"
Bridge Pier or Support	General "Fixed Object"
Bridge Rail	General "Fixed Object"
Culvert	Curb
Curb	Curb
Ditch	Curb
Embankment	Curb
Guardrail Face	Curb

Guardrail End	General "Fixed Object"
Concrete Traffic Barrier	General "Fixed Object"
Other Traffic Barrier	General "Fixed Object"
Tree (standing)	Tree
Utility Pole/Light Support	Pole
Traffic Sign Support	Pole
Traffic Signal Support	Pole
Other Post, Pole or Support	Pole
Fence	General "Fixed Object"
Mailbox	General "Fixed Object"
Other Fixed Object (wall, building, tunnel, etc.)	General "Fixed Object"
Unknown	General "Fixed Object"

Default vehicle symbols based on MMUCC vehicle movements

Vehicle Movement	Symbol as shown in the diagram legend
Movements Essentially Straight Ahead	Straight
Backing	Backing
Changing Lanes	Erratic
Overtaking/Passing	Overtaking
Turning Right	Right turn
Turning Left	Left turn
Making U-Turn	U-turn
Leaving Traffic Lane	Straight
Entering Traffic Lane	Straight
Slowing	Straight
Negotiating a Curve	Straight
Parked	Parked
Stopped in Traffic	Stopped
Other	Unknown
Unknown	Unknown

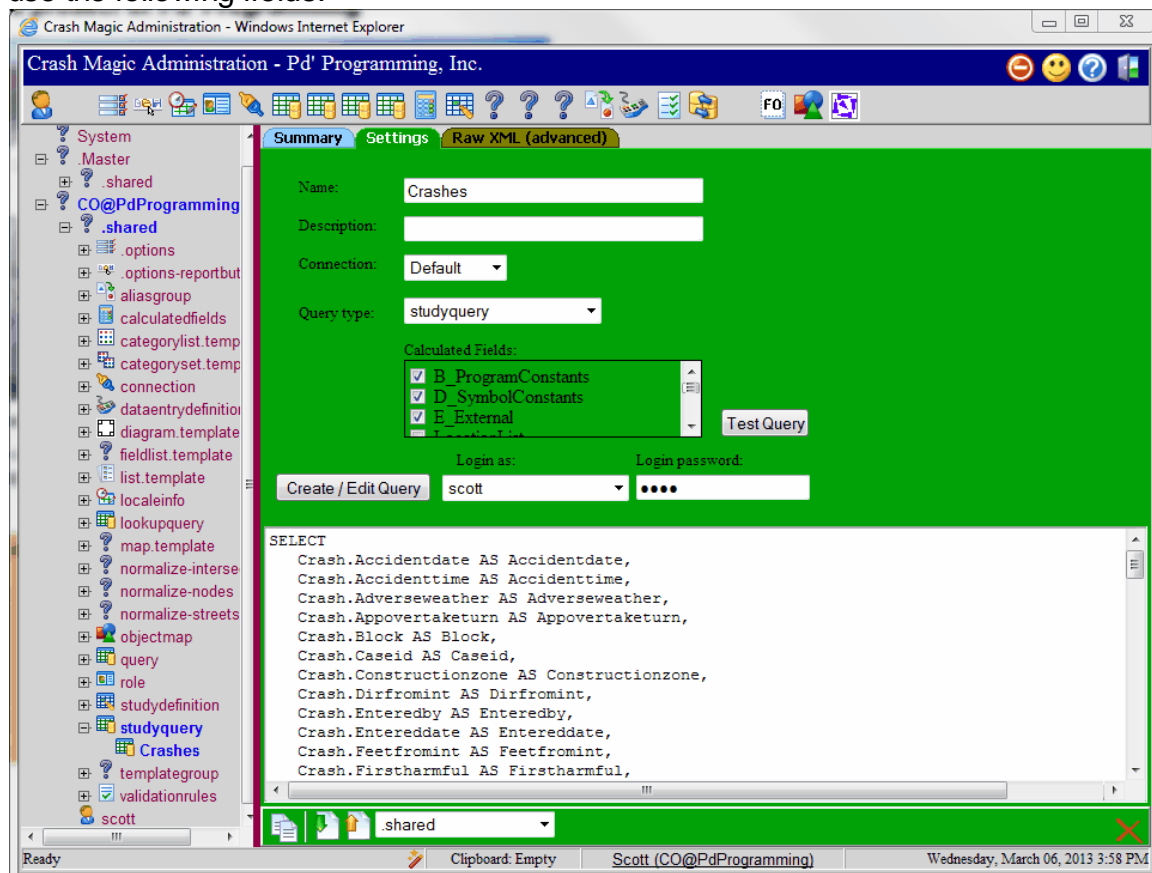
Default diagram symbols for Light Conditions, Crash Severity, and Alcohol Involvement based on MMUCC values.

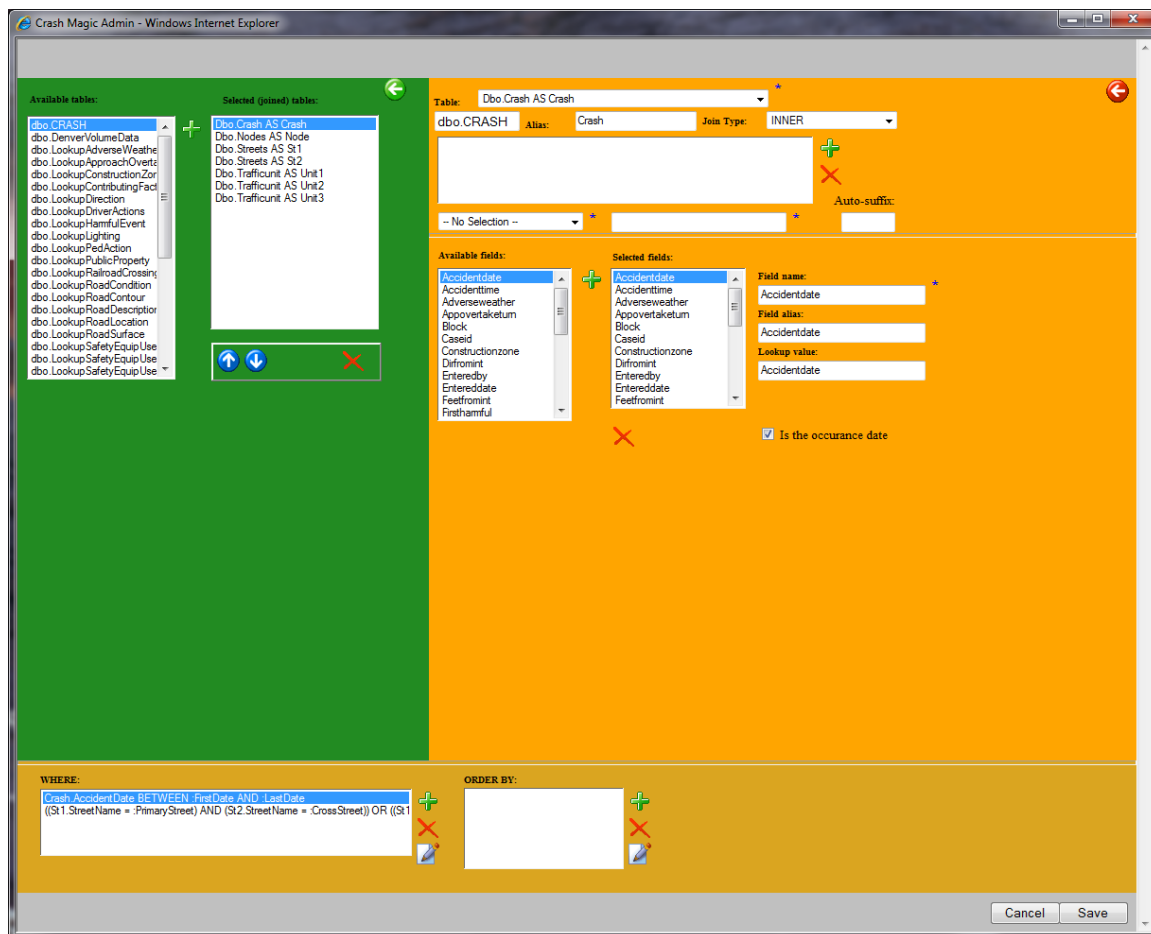
Item	Symbol
Light Condition	
Daylight	
Dawn	

Dusk	
Dark - Lighted	Nighttime
Dark - Not Lighted	Nighttime
Dark - Unknown Lighting	Nighttime
Other	
Unknown	
Crash Severity	
Fatal Injury	Fatality
Incapacitation Injury	Injury
Non-Incapacitating Injury	Injury
Possible Injury	Injury
Property-Damage-Only	
Unknown	
Alcohol Involvement	
No	
Yes	DUI
Unknown	

Queries

Queries extract the data required by Crash Magic from a collision database. All queries use the following fields:





Name: This is the name of the study query

Description: This is a description of the study query.

Connection: This is the [connection](#)^[227] that will be used by the query.

Query type: This defines how Crash Magic will view the query. The following query type options are available

- [lookupquery](#)^[275]
- [dependentlookupquery](#)^[276]
- [studyquery](#)^[276]

Test: This button is used to test the current query. 0=1 is added to the where clause and the query is run against the connection.

Calculated fields: This check box selects the [calculated fields](#)^[215] that can be used with the query.

Create / Edit Query: This button opens the query editor.

Login as: This is the Crash Magic user that will be used to issue the query.

Login password: This is the password provided to the Crash Magic login screen.

This query defines the fields a user can select when clicking on a collision graphic in a crash [diagram](#)^[20]. This query returns a single record with all the collision fields from the ClickOn query. The fields selected in this query should also match the fields used in the previous study queries. Crash Magic users will then be able to select the fields they wish to display in the ClickOn report of the crash diagram. The fields in the ClickOn query should be the same fields selected by the study queries.

The field or fields passed to this query from a diagram are defined in the [UniqueFields](#)^[221] of the P_Default Calculatedfields.

The example _UniqueFields:

```
"CRASH_NUM:integer=" + String(CASE_NUM,0) + ";CRASH_DATE:Date=" + CRASH_DATE
```

In this example the field CASE_NUM as an integer and CRASH_DATE as a date are being embedded into a collision diagram. When a user clicks on collision graphic these fields will be passed to the ClickOn query.

The ClickOn query must be set to receive the values being passed by the collision diagram.

Example where clause:

```
(( Crash.Case_Num = :CASE_NUM ) AND ( Crash.Crash_Date = :CRASH_DATE ))
```

In this example :CASE_NUM and :CRASH_DATE are passed to the ClickOn query, and the Crash table contains all of the collision records for the database. The ClickOn query then is set to return only the records where the :CASE_NUM passed matches the Crash.Case_Num field and :CRASH_DATE matches the Crash.Crash_Date field.

The General query is a catch-all for queries that do not (yet) have their own types such as "lookup" or "import lookup" or "click-on", etc.

Some common General query types are "streets" and "intersections" used by the normalizers.

DBVerify queries are created for clients that use [Database passthrough](#)^[195]. The DBVerify query attempts to query the database using the login and password provided when a user logs into Crash Magic. If Crash Magic does not have a Crash Magic account for the user or the query fails to access the database, the user will be denied access to Crash Magic.

An example query might be:

```
SELECT Crash.CrashID
FROM Crash
WHERE 1=0
```

The purpose of the query is not to retrieve data from the database, but to check that the user has access to the database. All DBVerify queries should have a where clause of 1=0 to ensure that no data is actually returned by the query.

This query is used to create a list of all the distinct intersections in a collision database that locates collisions by intersection. This query is used to populate the [Intersections Normalizer](#)²⁶⁴.

Example query:

```
SELECT
    St1.Streetname AS Streetname_St1,
    St2.Streetname AS Streetname_St2
FROM
    ( dbo.CRASH Crash
      INNER JOIN dbo.STREETS St1 ON ((St1.Streetid=Crash.Street1))
      INNER JOIN dbo.STREETS St2 ON ((St2.Streetid=Crash.Street2))
    )
UNION
SELECT
    St1.Streetname AS Streetname_St1,
    St2.Streetname AS Streetname_St2
FROM
    ( dbo.CRASH Crash
      INNER JOIN dbo.STREETS St1 ON ((St1.Streetid=Crash.Street2))
      INNER JOIN dbo.STREETS St2 ON ((St2.Streetid=Crash.Street1))
    )
```

In this example the collision database contains no table of intersections. So this query retrieves all of the intersections from all of the collisions in the database.


This query is used to create a list of all the streets in a client database. The streets query is only used for clients implementing the intersection study. This query and the streets normalizer are used to display the list of possible streets when a user starts to enter a street name in an intersection study, address or milepost study.

Example query:

```
SELECT
    St.Streetname AS Fullstreetname
FROM
    CITY.STREETS St
```

In this example the clients collision database contains a table called Streets that maintains a relationship the clients collision table.


Some clients do are not able to validate their street locations is this way, and may resort to querying all of the streets from their collision table.

Lookup queries  are used to query tables for converting database values to human readable values. For example a value can be assigned to the severity of a collision 1, 2, or 3. A look up query can query a table that converts these values to into "Property damage only", "Injury" or "Fatality".

Lookup queries are used by Crash Magic to change database codes to readable values in filters, ClickOn reports and other areas of the program. For example a numeric values of 1,2 or 3 can be stored to represent a lighting condition for each collision record. Crash Magic can use a lookup query to display what the values represent (Daylight, Dawn\Dusk or Dark) depending on the data .

Lookup query names should reflect what the query is looking up.

The following are the fields required for a lookup query:

DBFIELD: This is the name of the field that needs to be looked up. This value must match the [Lookup value](#)  for the field to be looked up for the study and ClickOn queries.

DBVALUE: This field contains the values that will be looked up.

DBLOOKUP: This field contains the information displayed to the Crash Magic user.

DBTABLE: Reserved for future use. This should be blank ("").

DBNAME: Reserved for future use. This should be blank ("").

In the example query below the TRAFFIC_LOOKUPS table contains lookups for all of the database fields. The table has three columns LU_CODE, LU_VALUE and LU_DESC.

```
(
SELECT
TRAFFIC_LOOKUPS.LU_CODE AS DBFIELD,
TRAFFIC_LOOKUPS.LU_VALUE AS DBVALUE,
TRAFFIC_LOOKUPS.LU_DESC AS DBLOOKUP,
'' AS DBNAME,
'' AS DBTABLE
FROM
TRAFFIC.TRAFFIC_LOOKUPS
```

Dependent Lookup queries can be used by [streets normalizers](#)^[266] to retrieve the readable text for a location based two fields. Like a lookup query that returns readable text for a single value, the Dependent lookup queries uses two values. Crash Magic stores the results of this query to display the text to the Crash Magic user

The following are the fields required for a lookup query:

DBNAME: Reserved for future use. This should be blank ("").

DBTABLE: Reserved for future use. This should be blank ("").

DBFIELD: This is the name of the field that needs to be looked up.

DBVALUE: This field contains the values that will be looked up.

DBPARENTVALUE: This field contains the parent value that will be used by the streets normalizer. This value is the Unique1 field from the streets normalizer.

DBLOOKUP: This field contains the information displayed to the Crash Magic user.

Crash Magic uses DBVALUE and DBPARENTVALUE to locate the DBLOOKUP field.

```
SELECT
    '' AS Dbname,
    '' AS Dbtable,
    'CITY_SECT_ID' AS DBFIELD,
    City.CITY_SECT_ID AS DBVALUE,
    Cnty.CNTY_ID AS DBPARENTVALUE,
    City.CITY_SECT_NM AS DBLOOKUP
FROM
    Dbo.CITY_SECT City
    INNER JOIN dbo.ST Cnty ON ((Cnty.City_Sect_Id=City.CITY_SECT_ID))
ORDER BY
    City.CITY_SECT_NM Asc
```

In this example the CITY_SECT_ID and the CNTY_ID fields are used to find the city name field, CITY_SECT_NM.

Study queries are used by [studies](#)^[12] and extract the collision records from the database for analysis. Study queries require a single record be returned for each

collision. The Crash Magic query editor allows fields from the query to be identified as normalized fields. Normalized fields are query fields that are mapped to the Crash Magic program. These fields tell the program when and where a collision occurred. Crash Magic uses the normalized fields to add where clauses information to the study query. Depending on the collision data in use some or all normalized fields will be used. [Study Definitions](#)²⁸⁰ are used to define which fields will be used when a new study is created.

Normalized fields are selected in the query editor using the Normalized field drop down menu.

Normalized field	Description
Date	The field that identifies the date of a collision
XCoordinate	X coordinate field of a collision
YCoordinate	Y coordinate field of a collision.
UserCID1	This is a unique field to identify a group of collisions or the UniqueNum1 field of the Crash Magic CID system table
UserCID2	This can be another user ID field or the UniqueNum2 field of the Crash Magic CID system table
cmUserGroupID	The UserGroupID field of the Crash Magic CID system table
cmUserID	The UserID field of the Crash Magic CID system table
cmProjectID	The ProjectID field of the Crash Magic CID system table
cmStudyID	The StudyID field of the Crash Magic CID system table
Address1	Block address field for the Primary Street. This label is used for collision data that is located by street and address.
Address2	Block address field for the Cross Street. This label is used for collision data that is located by street and address along with cross street and address.
Milepost	Milepost field where a collision occurred. This label is used for collision data that is located by route and milepost.
Route	Roadway field that identifies where a collision occurred. This label is used for collision data that is located by route and milepost.
RouteUnique1	Identifies the Route field is unique to an area. For example this could be used to identify the specific field that contains the county where the collision occurred.
RouteUnique2	Identifies the Route field is unique to an area within the area identified in RouteUnique1. For example this could be used to identify the specific field that contains the city within a county where the collision occurred.
PrimaryStreet	The field that contain the street the collision occurred on. This label is used for collision data that is located by street and address and/or street and cross street
PrimaryStreetUnique1	Identifies the PrimaryStreet field is unique to an area. For example this could be used to identify the specific field that

	contains the county where the collision occurred.
PrimaryStreetUnique2	Identifies the PrimaryStreet field is unique to an area within the area identified in PrimaryStreetUnique1. For example this could be used to identify the specific field that contains the city with in a county where the collision occurred.
CrossStreet	This field identifies the nearest intersecting street of a collision. This field is used for collision data that is located using an intersection of a primary street and cross street.
CrossStreetUnique1	Identifies the CrossStreet field is unique to an area.
CrossStreetUnique2	Identifies the CrossStreet field is unique to an area within the area identified in CrossStreetUnique1.

Clients that use the Map Magic selection tool will need to join the CID Crash Magic system table with the collision data. This generally requires that both the crash data and system tables reside on the same SQL server.

Study queries must have an order by clause. An order by clause will assure the database returns the records in the same order each time the query is performed. Without an order by clause the clients using collision diagrams may experience problems when moving collision graphics in a diagram. The id field or fields for the collision record should be selected for the order by clause. This field or fields are also used as the [CaseID](#)²¹⁵ field with in the calculated fields.

Where clauses should be avoided with study queries as they will be dynamically added to the query by the study definition.

If multiple study queries are required every effort should be used to use the same fields in the select statement of each study query and the [ClickOn](#)²⁷³ query.

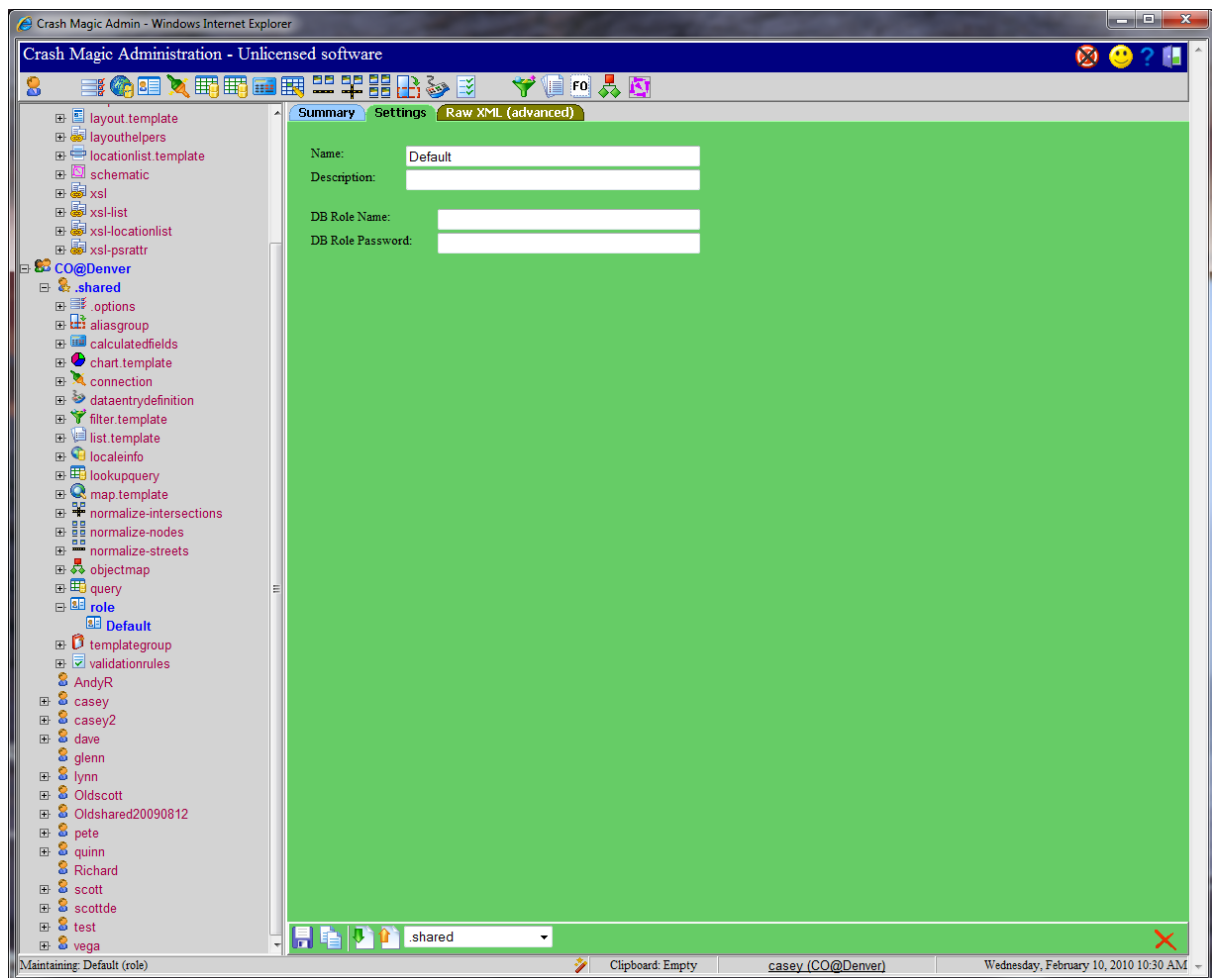
Each study definition must specify a study type and a query that it will use to gather data from the database. Each study type has parameters it requires. These parameters are obtained from the study query's normalized fields. The study types and required parameters are shown here.


Study type:	Description:	Required parameters:
All Data	The all data study gathers collisions by Date (DateTime) date range. It is not specific to a location.	
Address	The Address study gathers collisions by date range, route and address range.	Date (DateTime) PrimaryStreet (String) Address1 (Integer) Address2 (Integer) (optional) Caseld (String)
Case Id (CID)	The CID study provides the ability to query the database for case id's that have been populated in the crash	

Study type:	Description:	Required parameters:
	magic system tables. This query uses the user group id, user id, project id and the study id of the user that is logged in.	
Intersection	The Intersection study provides the ability to query the database by date range plus primary and cross streets. This is the standard study for urban areas and rural areas where mileposts don't exist.	tDate (DateTime) PrimaryStreet (String) CrossStreet (String)
Intersection corridor	The intersection corridor study provides the ability to query a sequence of intersections on a single street.	tDate (DateTime) PrimaryStreet (String) CrossStreet (String) _DistFromInt calculated field _DirFromInt calculated field
	This study type also provides the ability to specify parameters to include/exclude crashes occurring at intersections, off the corridor, and beyond the ends of the corridor.	
X,Y Coordinate (xyrect)	The xyrect study gathers crashes by prompting for a date range plus a rectangular area (X min/max and Y min/max values) that indicates a coordinate range from a GIS system.	Date (DateTime) XCoordinate (Float) (e.g. longitude) YCoordinate (Float) (e.g. latitude)
Node	The Node study uses a date range, and optionally a node identifier, up to two additional unique identifiers. These unique identifiers might specify a county, city, zip code, etc. The unique identifiers are only needed if multiple, non-unique instances of the node number occur in the database.	Date (DateTime) Node queries currently utilize the node field specified in the node normalizer. Node is expected to become a normalized field in the future.
Route + Milepost	The milepost study uses a date range plus a route name/id and milepost range.	Date (DateTime) Route (String) Milepost (Float)

Role (role)

The role button is used to define a database role that is required in connecting to a database. Not all databases have roles defined, but they are required for Crash Magic database connections.



To create a new roll click on the .shared user under the user group just created. Now click the role button  to create a role under the .shared user of the user group.

Name: The name of the role that will be used. If your database does not use roles we recommend that you change the name to Default, and leave all other fields blank.

Description: This is a description of the database role.

DB Role Name: This is the name of the role in the database Crash Magic will access.

DB Role Password: This is the password for the database role.

Study definition (studydefinition)

Study definitions determine the properties that a user will have when creating a new study. Study Definitions specify the query that will be used and the where clause that will be appended to the study.

Name: This is the name of the study definition

Description: This is a description of the study definition

Study Type: This is the type of study that will be used

- alldata - All Data
- cid - CID
- interection - Intersection
- intersectioncorridor - Intersection Corridor
- node - Node
- routemilepost - Route Milepost
- streetaddress - Street Address
- usercid1 - User CID
- usercid2 - User CID
- xyrect - XY Rectangle

Query: This is the [study query](#)²⁷⁶ that will be used

Normalizer: This is the normalizer that will be used by the study definition. Collision data that can use node and intersections to location collisions should contain a node and intersection normalizers with the same name.

All data study definitions are used for studies that only contain date ranges. This study will append a where clause to the query selected for this study. For the All Data study to work a query must be used that identifies the following normalized field:

- [Date](#)²⁷⁷

Example where clause added to the all data query specified in the study definition

WHERE

```
(( Crash.Accidentdate BETWEEN '1/1/2007' AND '12/31/2007 11:59:59 PM' ))
```

In this example the Accidentdate field from the study query was labeled as the Normalized date field. Crash Magic add the date information based on the user dates selected in the All Data study.





Normalized fields required to produce Intersection High Crash Location lists with this study definition:

- [PrimaryStreet](#)²⁷⁷
- [CrossStreet](#)²⁷⁸

Optional fields that may be used depending on collision data:

- [PrimaryStreetUnique1](#)²⁷⁷
- [PrimaryStreetUnique2](#)²⁷⁸
- [CrossStreetUnique1](#)²⁷⁸
- [CrossStreetUnique2](#)²⁷⁸

The CID study stands for Case ID List. A CID study is used by systems that pass a list of crash IDs into Crash Magic for analysis. When a web page or application posts crash IDs to Crash Magic, these IDs are stored in the Crash Magic CID system table(<table prefix>CID) along with the Crash Magic user group ID, user ID, project ID, and study ID of the current user that the data was posted against. The study query selected for the CID study definition joins the UniqueNum1 field of the CID system table to the unique collision ID of a clients collision table. The following fields must be labeled in the selected study query for this study definition to work:

- [cmUserGroupID](#) 
- [cmUserID](#) 
- [cmProjectID](#) 
- [cmStudyID](#) 

Example Join to the Crash Magic CID system table

```
FROM
      ( CMO.cmCID CID
      LEFT JOIN Dbo.Crash Cr ON (Cr.CrashReport=CID.UniqueNum1) )
```

In this example the UniqueNum1 field of the cmCID Crash Magic system table is joined to the CrashReport field of the clients Crash table.





Example where clause added to the all data query specified in the study definition

```
WHERE
(CID.UserGroupID = :cmUserGroupID)
AND(CID.UserID = :cmUserID)
AND(CID.ProjectID = :cmProjectID)
AND(CID.StudyID = :cmStudyID)
```

Intersection study definitions are used for studies that locate collisions by a primary street and cross street. This study will append a where clause to the query selected for this study. For the Intersection study to work a query must be used that identifies the following normalized field:

- [Date](#) 
- [PrimaryStreet](#) 
- [CrossStreet](#) 

Optional fields that can be identified in the query:

- [PrimaryStreetUnique1](#) 
- [PrimaryStreetUnique2](#) 
- [CrossStreetUnique1](#) 
- [CrossStreetUnique2](#) 

The intersection study will swap the streets as "PrimaryStreet and CrossStreet" and "CrossStreet and PrimaryStreet" in the where clause to obtain all the collisions for a given intersection location.

Example where clause added to the intersection query specified in the study definition:

```
WHERE
(( Crash.Crash_Dt BETWEEN '1/1/2008' AND '12/31/2008 11:59:59 PM'))
AND( (
(
(
( Crash.Cnty_Id = 26)
AND
(
( Crash.Cnty_Id = 26)
AND
(
( Crash.City_Sect_Id = 249)
AND
(
( Crash.City_Sect_Id = 249)
AND
(
(( Crash.St_Full_Nm = 'NW BROADWAY') AND ( Crash.Isect_St_Full_Nm = 'NW MAIN ST')) OR
(( Crash.St_Full_Nm = 'NW MAIN ST') AND ( Crash.Isect_St_Full_Nm = 'NW BROADWAY'))
)
)
)
)
)
)
)
)
)
```

In this example the **Crash_Dt** field from the **Crash** table was labeled as the **Normalized date field**. The **Cnty_Id** field from the **Crash** table has been labeled as the **PrimaryStreetUnique1** and **CrossStreetUnique1** normalized fields. The field **City_Sect_ID** from the **Crash** table has been labeled as **PrimaryStreetUnique2** and **CrossStreetUnique2** normalized fields. The field **St_Full_Nm** from the **Crash** table has been labeled as the **PrimaryStreet** name, and the **Isect_St_Full_Nm** from the **Crash** table has been labeled as the **CrossStreet** name.

The previous example uses a county and city field to ensure this location is unique to a state area.

To Set Up Corridor Dynamic Diagrams:

- * Database:
 - * Create OrderedStreets table.
 - * Create pdg GIS services tables.
 - * Provide read/write access to both OrderedStreets and pdg tables.
- * P2_Default must have _Latitude and _Longitude (As number. If lat/long fields are string use AsNumber() in expression)
 - * Create Diagram schematic PsrAttr, named "DynamicIntersection"
 - * Create GIS PsrAttr:
 - * Must configure both a geocode server and centerline server to enable ordered streets.

* Must create pdg tables to enable database caching of GIS data for performance reasons.

* Create TableDef PsrAttr, named "OrderedStreets", point to OrderedStreets table, set reader/writer as appropriate.

* Create Intersection Normalizer PsrAttr:

* Named "Default"

* Normalize Streets: Default.

* Distinct intersections query: Intersections.

* Set Primary and Cross street field names.

* Select OrderedStreets table.

* check both Hidden and Visible.

* On Dependent data tab: select OrderedStreets.

* If no intersection corridor study definition, create Study Definition PsrAttr, named "Intersection Corridor". Study Query Type: intersectioncorridor. Query: Crashes. Normalizer: Default.

Node study definitions are used for studies that only contain date ranges. This study will append a where clause to the query selected for this study. For the Node study to work a query must be used that identifies the following normalized field:

- [Date](#) 

The node study will determine the node field from the node normalizer, and therefore it does not need to be labeled in the query unlike other studies.

Route Milepost study definitions are used for studies that locate collisions by a route and milepost. This study will append a where clause to the query selected for this study.

- [Date](#) 
- [Route](#) 
- [Milepost](#) 




Optional fields that can be identified in the query:

- [RouteUnique1](#) 
- [RouteUnique2](#) 

The Street Address study definitions are used for studies that locate collisions by a primary street and address block. This study will append a where clause to the query selected for this study.

- [Date](#) 
- [PrimaryStreet](#) 
- [Address1](#) 

Optional fields that can be identified in the query:

- [Address2](#) 
- [PrimaryStreetUnique1](#) 
- [PrimaryStreetUnique2](#) 

The UserCID1 study definitions are used for studies that locate collisions by a single ID. This study allows Crash Magic to identify a group of collisions based on a single ID field. For the UserCID1 study to work a query must be used that identifies the following normalized field:



- [UserCID1](#) 

Example where clause added to the User CID 1 query specified in the study definition.

```
WHERE ((Crash.RequestID = 2895))
```

In this example RequestID was labeled as the UserCID1 field in the study query. Crash Magic then passed the number 2895 entered by the user.

The UserCID2 study definitions are used for studies that locate collisions by two IDs. This study allows Crash Magic to identify a group of collisions based on a two ID fields. For the UserCID2 study to work a query must be used that identifies the following normalized fields:

- [UserCID1](#) 
- [UserCID2](#) 


Example where clause added to the User CID 1 query specified in the study definition.

```
WHERE ((Crash.RequestID = 2895))
```

```
AND(( User.ItemNum = 20 ))
```

In this example RequestID was labeled as the UserCID1 field and ItemNum was labeled as the UserCID2 field in the study query. Crash Magic then passed from the users study the number 2895 for the UserCID1 field and 20 for UserCID2 field.

The XY Rectangle study definitions are used for studies that only contain GIS data. This study will append a where clause to the query selected for this study. For the XY Rectangle study to work a query must be used that identifies the following normalized field:

- [Date](#) 
- XCoordinate
- YCoordinate

Utility - Text lines (textlines)

Utility - Text lines is a general PSRattr that can store any text. This section contains documentation for the currently defined uses.

Crash Magic was designed to be able to import xml files. Clients that will be using the import feature of Crash Magic will need to convert the data data being imported into an xml file format. Importing data from a text source may require that a user create a schema.ini file to define the file being imported. The ini file is created in the process of setting up a Windows data source. The ini file must be in the same directory as the text file the is being imported to work. The following steps show how to create a schema.ini for a text file with comma separated columns and column headers for each column on a windows 7 machine.

Open your ODBC data source administrator

1. Click on the windows (start) button and select the Control Panel.
2. If in Category View then click on System and Security.
3. Click on Administrative Tools.
4. Click on Data Sources (ODBC) to open the ODBC Data Source Administrator window.

Locate the directory of the file

1. With the User DSN tab selected on the ODBC Data Source Administrator window open click the Add button.
2. Select the Microsoft access Text Driver(*.txt, *.csv)
3. Click the Finish button to open the ODBC Text setup window.

Select a directory where the text file exists

1. With the ODBC Text setup window still open, uncheck the Use Current Directory.
2. Click the Select Directory... button.
3. Browse to the file that you would like to create a schema.ini file for.

Select text extension for importing schema.ini will be created for

1. With the ODBC Text setup window still open select the *.txt from the Extensions List.

Define a format for the file

1. With the ODBC Text setup window still open click the Define Format.. button
2. In the Define Text Format window select your text file.
3. Click the Column Name Header check box.
4. Select the format of the text file.
5. Click the Guess button to populate the Columns.
6. Click the Ok button when done to create the file.
7. Click the Cancel button on previous windows as only the ini file is needed.

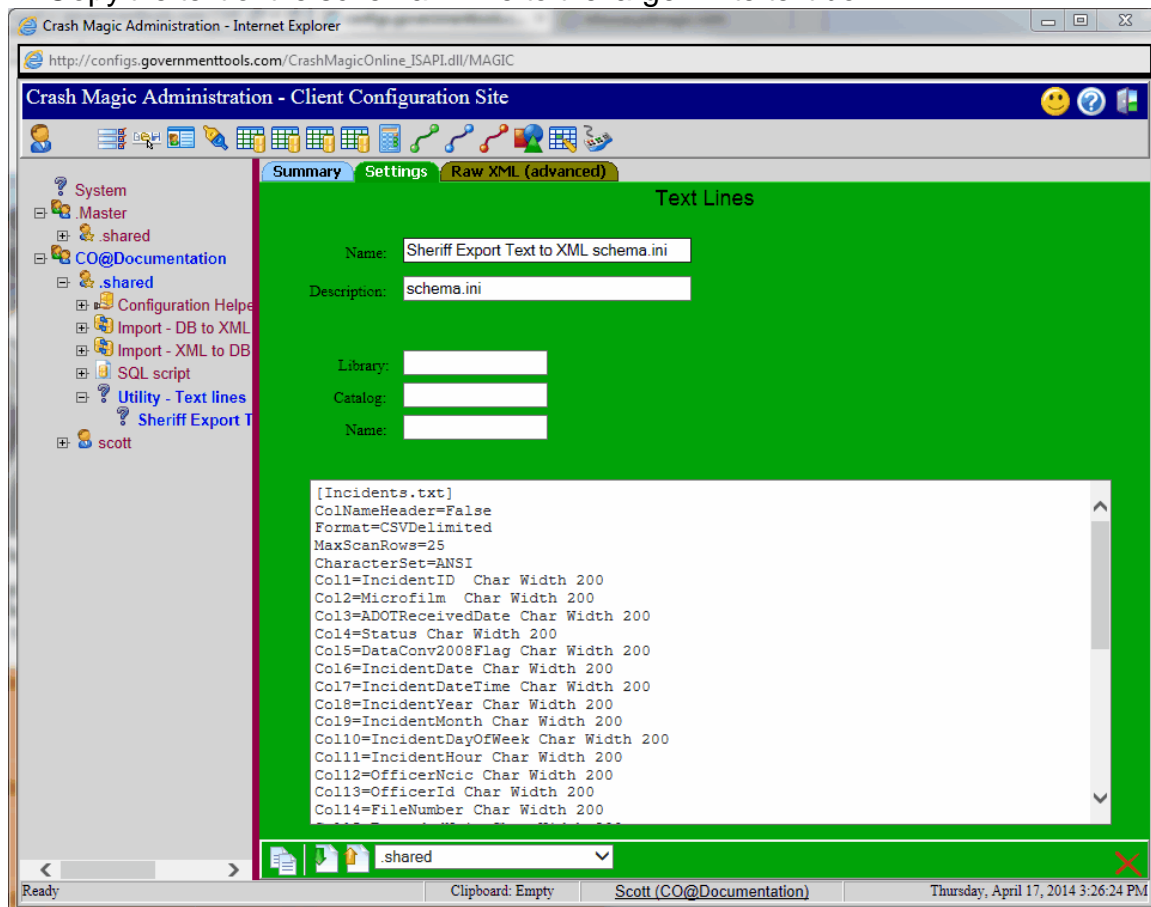
The following steps created a schema.ini file in the directory with your text file. Once the file is created it will need to be added to the Crash Magic configuration.

Create the Utility - Text lines PSRattr

1. Log into Crash Magic as the Group Administrator.
2. Click on the .shared user under the user group you have logged into.
3. Select the green Settings tab.
4. Click on the drop down menu in the Utility functions box and select Utility - Text lines (textlines).
5. Click the Create new PSRattr button.

Add the schema.ini information:

1. Crash Magic should now display the green settings tab of the Utility - Text lines that was created from the previous task.
2. Change the Name box to the <data source name> Text to XML schema.ini.
3. Enter a description of schema.ini.
4. Copy the text of the schema.ini file to the large white text box.



5. Click the the blue Summary tab to save the settings.

User defined buttons are resources that cause buttons with custom URL links to appear in the project, study or report toolbar. (below the report)

There are three different user defined button resources. They are identified by being a "Utility - Text lines" resource with specific names.

- UserProjectButtons - the buttons specified in this resource will be show on project, study and report pages. They are the most general and can only access project information. They will not have knowledge of the current study or report.
- UserStudyButtons - the buttons specified in this resource will be show on study and report pages. They are the most common as they can access study information as well as project information. They will not have knowledge of the current report.
- UserReportButtons - the buttons specified in this resource will be show only on report pages. They can access report information as well as project and study information.

All three of these resources share the same format. The only difference is where they are displayed. The format is delimited. For technical reasons, the values are delimited with semi-colon characters (;). If a value contains a semi-colon, it must be quoted with pipe characters (|).

The values on each line must be the following and must be in this order:

1. URL - this is the desired URL including any parameters.
2. Hint - shown in a "tool tip" when mouse-over occurs
3. Target - can be used to open a new tab/window or use an existing one
4. Image - a 24x24 pixel image to be used for the button

Each value will be parsed by Crash Magic before use. So any static text (literals) must be enclosed in quotes.

Here is a simple example that creates a button that opens the Pd' Programming web site when it is clicked:

```
"https://www.pdmagic.com";"Open the Pd' Programming web site";"PdSite";"https://www.governmenttool
```

However, in most cases the purpose will be to link to locations that address a dynamic path or take parameters, probably related to the current project, study or report.

```
"http://www.pdmagic.com/" + URLPathEncode( Project.Name.ASCII( " ", " " ) ) + "/" + URLPathEncode( Study
```

or

```
"http://www.pdmagic.com/SendDataHere/?Project=" + URLParamEncode( Project.Name.ASCII( " ", " " ) ) + "&S  
"Open Study for editing";"PdSite";"Editor"
```

Some notes:

- The expressions can utilize any of the [filter functions](#)^[80] available in the program.
- When using a variable, such as the current project name or study name or such, be sure to wrap it in URLPathEncode() to make sure a legal URL is created.
- When using a parameter, use URLParamEncode().

- The image value may point to any image available by URL. It will be scaled to 24x24 pixels. Crash Magic has a plethora of icons to choose from if there is not a suitable custom image available. An index of the image is being prepared.

Validation Rules (validationrules)

Crash Magic allows validation rules to be created for data entry. Two types of validations exist Errors and Warnings. An Error will not allow the record to be saved. A warning will alert the user to a problem, but allow the record to be saved.

Common validation rules set for Data Entry.

Fields for required information:

- Error - Primary Street the collision occurred on.
- Error - Cross Street, Block or Milepost information is missing.
- Error - Date of collision.
- Warning - Time has not been entered.

Fields that require another field:

- Warning - Vehicle total does not match actual vehicle information(Three total vehicles in the collision, but only one vehicle entered).
- Warning - Distance and Direction from intersection(If direction is entered there must be a distance).
- Warning - Vehicle direction and maneuver(If a maneuver is entered then a direction must also be given).
- Error - Crash severity does not match number injured or fatal(Collision severity is fatal, but number killed is 0).
- Warning - Pedestrian collision, but pedestrian not entered.
- Warning - Cyclist collision, but cyclist not entered.

Fields outside of expected range:

- Warning - Speed limit greater than 75.
- Warning - Speed of vehicle greater than 80.
- Warning - Driver age greater than 95.
- Warning - Driver age less than 10.

Current XML definition for data entry validation rules. The purpose of this section is to document the XML structures used by the data entry validation rule. Clients should not alter the XML directly. The XML defines the validation rules used by the client to ensure data is valid on entry.

Defining a validation rule

```
<ValidationRule>
```

```

    <Expression>Null(AccidentFormDate.value)</Expression>
    <Explanation>Date is blank</Explanation>
    <Level>Error</Level>
    <RelatedFields/>
  </ValidationRule>

```

Example validation rule

Expression - A valid Crash Magic expression that resolves to true or false. Expressions that evaluate to true will cause the validation rule to fire. In the preceding example the value of the data entry field AccidentFormDate is checked for null.

Validation rules use standard filter [functions](#)^[80].

Explanation - The reason displayed to the user on why the validation rule was triggered.

Level - This can be set to Error or Warning. Error will prevent the record from being saved. Warnings will only display the validation rule to the user.

RelatedFields - This field is reserved for future use.

```

<pdroot>
  <ValidationRules>
    <ValidationRule>
      <Expression>Null(AccidentFormDate.value)</Expression>
      <Explanation>Date is blank</Explanation>
      <Level>Error</Level>
      <RelatedFields/>
    </ValidationRule>
    <ValidationRule>
      <Expression>Null(AccidentFormTime.value)
</Expression>
      <Explanation>Time is blank</Explanation>
      <Level>Error</Level>
      <RelatedFields/>
    </ValidationRule>
    <ValidationRule>
      <Expression>NumberOfFatalities.value .GT. 3
</Expression>
      <Explanation>Number of fatalities is greater than 3</Explanation>
      <Level>Warning</Level>
      <RelatedFields/>
    </ValidationRule>
    <ValidationRule>
      <Expression>Null(Node.value)
</Expression>
      <Explanation>Node is blank</Explanation>
      <Level>Error</Level>
      <RelatedFields/>
    </ValidationRule>
  </ValidationRules>

```

```

        <Expression>(FormUnit1Type.value .EQ. 4) .AND.
(AppOvertakeTurn.value .NEQ. 3)

</Expression>
        <Explanation>C. App/Overtake should be N/A when unit 1 is
Pedestrian</Explanation>
        <Level>Warning</Level>
        <RelatedFields/>
    </ValidationRule>
    <ValidationRule>
        <Expression>NumberInjured.value .GT. 5

</Expression>
        <Explanation>Number of injuries is greater than 5</Explanation>
        <Level>Warning</Level>
        <RelatedFields/>
    </ValidationRule>
    <ValidationRule>
        <Expression>Form1SpeedLimit.value .GT. 50
</Expression>
        <Explanation>Speed limit for Unit 1 is greater than
50</Explanation>
        <Level>Warning</Level>
        <RelatedFields/>
    </ValidationRule>
    <ValidationRule>
        <Expression>Form2SpeedLimit.value .GT. 50
</Expression>
        <Explanation>Speed limit for Unit 2 is greater than
50</Explanation>
        <Level>Warning</Level>
        <RelatedFields/>
    </ValidationRule>
    <ValidationRule>
        <Expression>Form1Speed.value .GT. 60
</Expression>
        <Explanation>The speed for Unit 1 is greater than
60</Explanation>
        <Level>Warning</Level>
        <RelatedFields/>
    </ValidationRule>
    <ValidationRule>
        <Expression>Form2Speed.value .GT. 60
</Expression>
        <Explanation>The speed for Unit 2 is greater than
60</Explanation>
        <Level>Warning</Level>

```

```

        <RelatedFields/>
    </ValidationRule>
    <ValidationRule>
        <Expression>(FormUnit1Type.value .NEQ. 4) .AND.
(IsNull(FormPed1Action.Value) .EQ. FALSE)
</Expression>
        <Explanation>The Pedestrian 1 Act field has a value, but Unit 1 is
not a pedestrian</Explanation>
        <Level>Warning</Level>
        <RelatedFields/>
    </ValidationRule>
    <ValidationRule>
        <Expression>(FormUnit2Type.value .NEQ. 4) .AND.
(IsNull(FormPed2Action.Value) .EQ. FALSE)
</Expression>
        <Explanation>The Pedestrian 2 Act field has a value, but Unit 2 is
not a pedestrian</Explanation>
        <Level>Warning</Level>
        <RelatedFields/>
    </ValidationRule>
    <ValidationRule>
        <Expression>Form1Age.value .GT. 95

</Expression>
        <Explanation>Unit 1 age is greater than 95</Explanation>
        <Level>Warning</Level>
        <RelatedFields/>
    </ValidationRule>
    <ValidationRule>
        <Expression>Form2Age.value .GT. 95

</Expression>
        <Explanation>Unit 2 age is greater than 95</Explanation>
        <Level>Warning</Level>
        <RelatedFields/>
    </ValidationRule>
</ValidationRules>
<Settings/>
</pdroot>

```

XSL-FO document (fotemplate)

FO Templates are used by Crash Magic to define printed reports. All documents are XML and use the W3C standard of XSL-FO to describe the page.

There are special Insert tags in the document that describe places that information is going to be filled in by Crash Magic. This allows the same template to be used for a number of different reports.

Insert Tag:

Name: A descriptive name of the region. This is used in the layout editor to select a region.

ForUseIn: Tells crash magic what kind of data can be inserted in place of this tag. Current values are: FO and ASCII.

Height: Optional. Is applied to the report element to ensure the correct size is met, and to prevent flowing to the next page.

Width: Optional. Is applied to the report element to ensure the correct size is met, and to prevent flowing to the next page.

Warning! Insert tags must be in the namespace:
<http://www.pdmagic.com/pdEl/1.0>

XSL-FO is a W3C specification. Information is available at the W3C website (<http://www.w3.org/>)

Crash Magic FO templates support an additional namespace (xmlns:pdEl="http://www.pdmagic.com/pdEl/1.0">) This namespace is used to insert dynamic data into FO templates for rendering to pdf.

Standard XSL-FO content is wrapped inside pdroot and pdfotemplate tags.

```
<pdroot>
  <pdfotemplate>
    <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format"
      xmlns:pdEl="http://www.pdmagic.com/pdEl/1.0">
      <fo:layout-master-set>
        <fo:simple-page-master page-height="8.5in" page-width="11in" margin-top="0in" margin-bottom="0in" margin-left="0in" margin-right="0in">
          <fo:region-body region-name="xsl-region-body" margin-top="0in" margin-bottom="0in" margin-left="0in" margin-right="0in">
            <fo:region-before region-name="xsl-region-before" extent="0.25in" margin-top="0in" margin-bottom="0in" margin-left="0in" margin-right="0in">
              <fo:region-start region-name="xsl-region-start" extent="0in"/>
              <fo:region-end region-name="xsl-region-end" extent="0in"/>
            </fo:region-before>
          </fo:region-body>
        </fo:simple-page-master>
      </fo:layout-master-set>
      <fo:page-sequence master-reference="only">
        <fo:flow flow-name="xsl-region-body">
          <fo:block-container width="100%">
            <fo:block text-align="center">
              <fo:instream-foreign-object border="solid black 1px" display="block" height="850px" width="1000px">
                <svg xml:space="preserve" viewBox="0 0 850 1000">
                  <g>
```

```

<defs>
  <linearGradient id="topGrad" x1="0%" y1="0%" x2="100%" y2="100%" spreadMethod="pad">
    <stop offset="0%" stop-color="#FFFFFF"/>
    <stop offset="100%" stop-color="rgb( 50, 118, 210)"/>
  </linearGradient>
</defs>

<rect x="0" y="0" width="850" height="45" style="fill: url(#topGrad); stroke: black;"/>
<text y="22" style="font-family: sans-serif; font-size: 18pt; text-anchor: end;" x="840">
  <pdEl:Insert ForUseIn="ASCII" Name="
</text>
<text y="42" style="font-family: sans-serif; font-size: 14pt; text-anchor: end;" x="840">
  <pdEl:Insert ForUseIn="ASCII" Name="
</text>
</g>

</svg>

</fo:instream-foreign-object>
</fo:block>
</fo:block-container>
<fo:block>
  <fo:table width="10.5in">
    <fo:table-body>
      <fo:table-row>
        <fo:table-cell>
          <fo:block text-align="left">
            <pdEl:Insert ForUseIn=
          </fo:block>
        </fo:table-cell>
        <fo:table-cell>
          <fo:block text-align="right">
            <pdEl:Insert ForUseIn=
          </fo:block>
        </fo:table-cell>
      </fo:table-row>
    </fo:table-body>
  </fo:table>
</fo:block>
<fo:block text-align="center">
  <pdEl:Insert ForUseIn="FO" Name="Main" Height="6.5in" Wid
</fo:block>
</fo:flow>
</fo:page-sequence>
</fo:root>
</pdfotemplate>
<Settings/>
</pdroot>

```

XSLT - General (xsl)

XSL is used in several places in Crash Magic. Each XSL attribute has an object type appended to the end. This object type defines the type of report that the XSL was designed for. Each XSL is wrapped in a pdxsl tag with an OutputType attribute. This allows multiple XSLs, with different output options to be grouped in the same attribute.

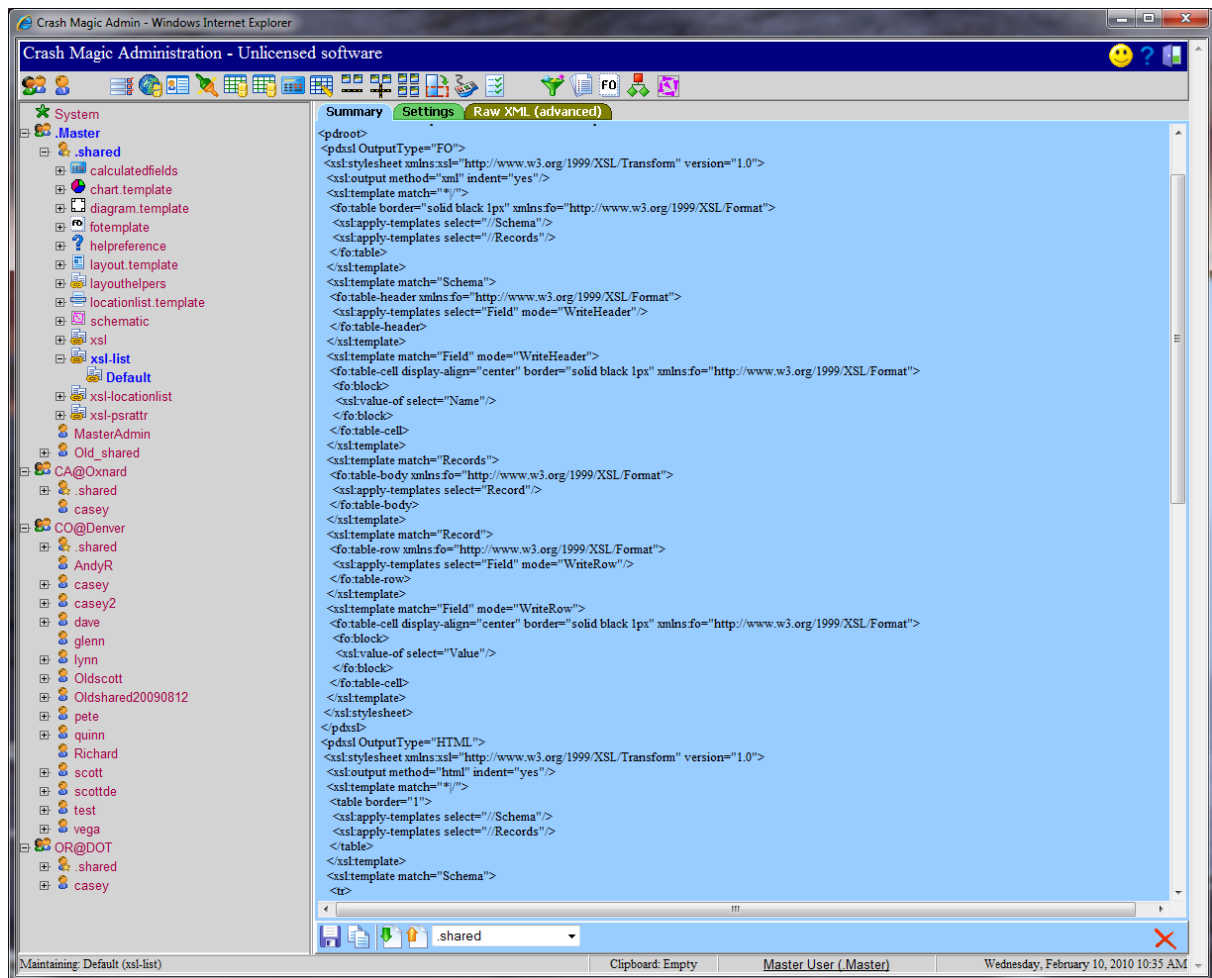
OutputType:

- FO

- HTML
- CSV

Types of XSL:

- xsl-fieldlist
- xsl-list
- xsl-locationlist
- xsl-psrattr



```

<pdroot>
  <pdxsl OutputType="FO">
    <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
      <xsl:output method="xml" indent="yes"/>
      <xsl:template match="*" />
        <fo:table border="solid black 1px" xmlns:fo="http://www.w3.org/1999/XSL/Format">
          <xsl:apply-templates select="//Schema"/>
          <xsl:apply-templates select="//Records"/>
        </fo:table>
      </xsl:template>
      <xsl:template match="Schema">

```

```

        <fo:table-header xmlns:fo="http://www.w3.org/1999/XSL/Format">
            <xsl:apply-templates select="Field" mode="WriteHeader"/>
        </fo:table-header>
    </xsl:template>
    <xsl:template match="Field" mode="WriteHeader">
        <fo:table-cell display-align="center" border="solid black 1px" xmlns:f
            <fo:block>
                <xsl:value-of select="Name"/>
            </fo:block>
        </fo:table-cell>
    </xsl:template>
    <xsl:template match="Records">
        <fo:table-body xmlns:fo="http://www.w3.org/1999/XSL/Format">
            <xsl:apply-templates select="Record"/>
        </fo:table-body>
    </xsl:template>
    <xsl:template match="Record">
        <fo:table-row xmlns:fo="http://www.w3.org/1999/XSL/Format">
            <xsl:apply-templates select="Field" mode="WriteRow"/>
        </fo:table-row>
    </xsl:template>
    <xsl:template match="Field" mode="WriteRow">
        <fo:table-cell display-align="center" border="solid black 1px" xmlns:f
            <fo:block>
                <xsl:value-of select="Value"/>
            </fo:block>
        </fo:table-cell>
    </xsl:template>
</xsl:stylesheet>
</pdxsl>
<pdxsl OutputType="HTML">
    <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
        <xsl:output method="html" indent="yes"/>
        <xsl:template match="*" />
        <table border="1">
            <xsl:apply-templates select="//Schema"/>
            <xsl:apply-templates select="//Records"/>
        </table>
    </xsl:template>
    <xsl:template match="Schema">
        <tr>
            <xsl:apply-templates select="Field" mode="WriteHeader"/>
        </tr>
    </xsl:template>
    <xsl:template match="Field" mode="WriteHeader">
        <td>
            <xsl:value-of select="Name"/>
        </td>
    </xsl:template>
    <xsl:template match="Records">
        <xsl:apply-templates select="Record"/>
    </xsl:template>
    <xsl:template match="Record">
        <tr>
            <xsl:apply-templates select="Field" mode="WriteRow"/>
        </tr>
    </xsl:template>
    <xsl:template match="Field" mode="WriteRow">
        <td>
            <xsl:value-of select="Value"/>
        </td>
    </xsl:template>

```

```

        </td>
    </xsl:template>
</xsl:stylesheet>
</pdxsl>
<pdxsl OutputType="CSV">
    <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
        <xsl:output method="text"/>
        <xsl:template match="*" />
        <xsl:apply-templates select="//Records"/>
    </xsl:template>
    <xsl:template match="Records">
        <xsl:apply-templates select="Record"/>
    </xsl:template>
    <xsl:template match="Record">
        <xsl:for-each select="Field">
            <xsl:value-of select="Value"/>
            <xsl:if test="position() != last()">
                <xsl:value-of select="', '"/>
            </xsl:if>
        </xsl:for-each>
        <xsl:value-of select=" ' " />
    </xsl:template>
</xsl:stylesheet>
</pdxsl>
<Settings/>
</pdroot>

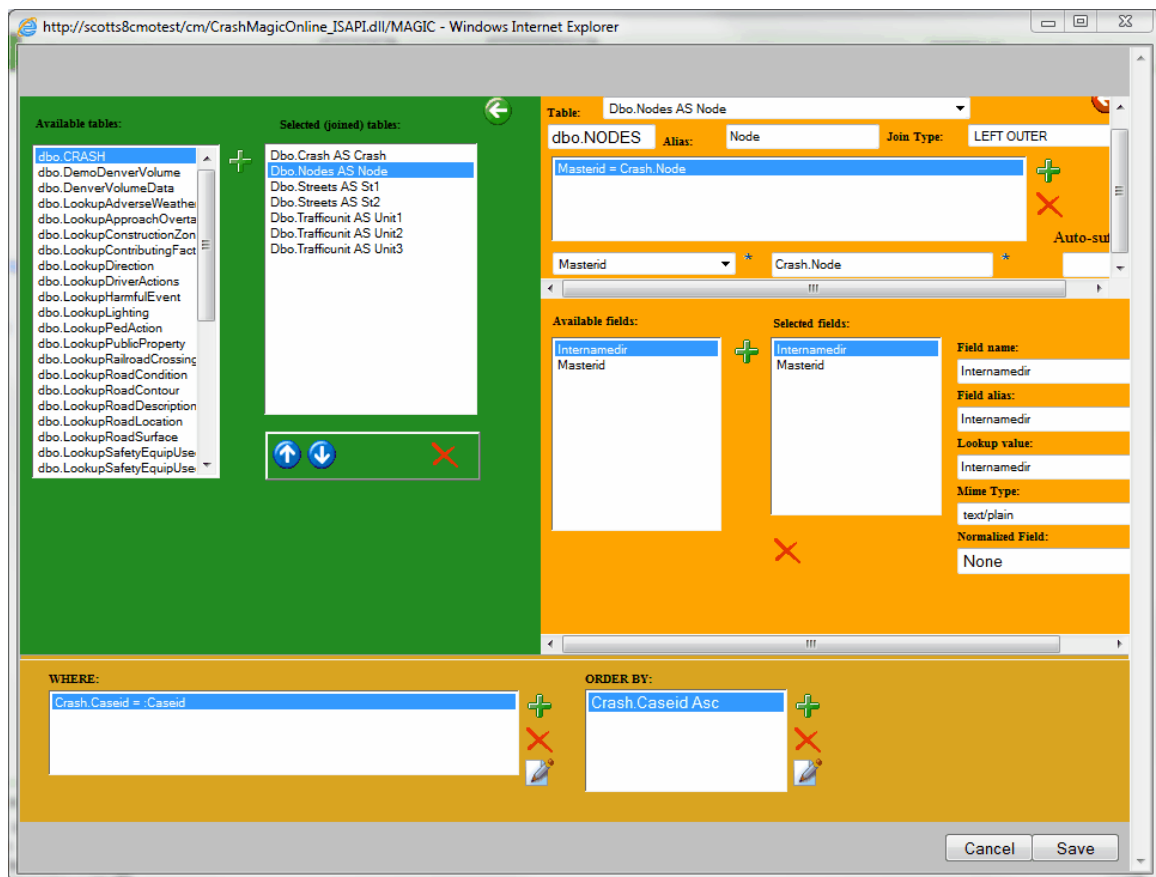
```

Query Editor

The query editor allows the user to create queries for Crash Magic. Clients using this editor must have a strong background in SQL and knowledge of the database in use. Studies queries are designed to create a flat collision record. This means that each row returned by the query will only contain one collision record. A query that returns a collision record in more than one row will cause users to receive duplicate record errors in Crash Magic.

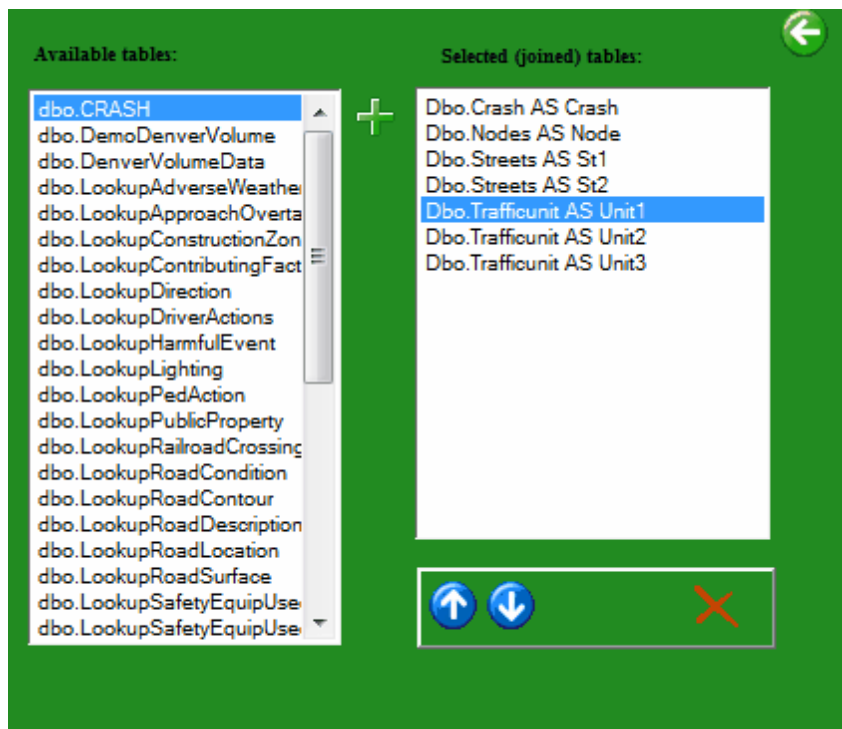
The editor is divided up into four sections:

- Table selector
- Table joins
- Query fields
- Where and order by clauses








The query editor is used for the creation of all Crash Magic queries

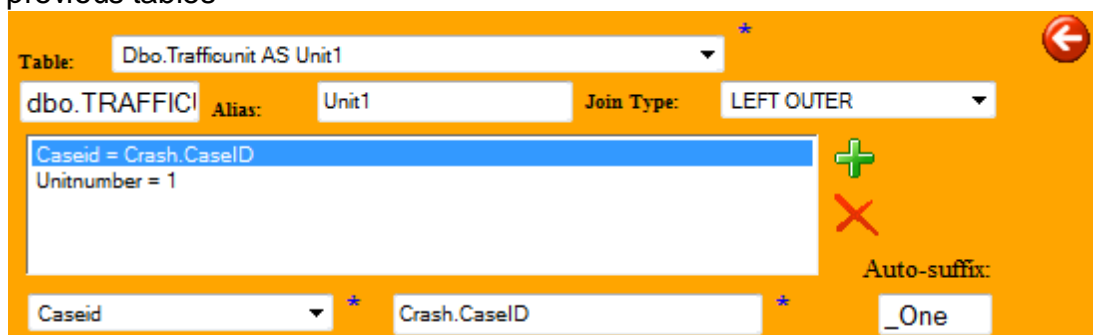
The Table selector is used to select the tables that will be used in a query and specify the current join condition displayed in the Table joins section of the window.



In this example the Streets and Trafficunit tables have been selected several times

- Available tables - Tables available for use in the query
- Selected (joined) tables - Tables currently selected in the query. These tables will be displayed in the from clause of the query.
-  - Hides the tables section of the query editor
-  - Adds the highlighted table from the Available tables list to the Selected (joined) tables list (A table can be selected more than once to create self joins)
-  - Move selected table up in the Selected (joined) tables list
-  - Move selected table down in the Selected (joined) tables list
-  - Remove selected table from the Selected (joined) tables list



The Table Join section is used to tell how the selected table will be joined to the previous tables

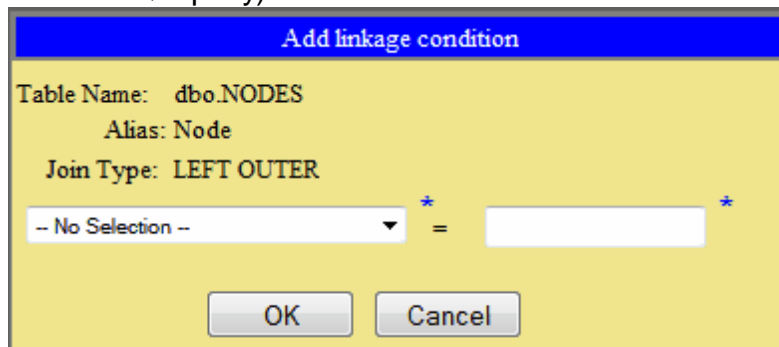


In this example the table alias is Unit1.

Example Join created


```
LEFT JOIN dbo.TRAFFICUNIT Unit1 ON ((Unit1.Caseid=Crash.CaseID) AND (Unit1.UnitNumber=1))
```

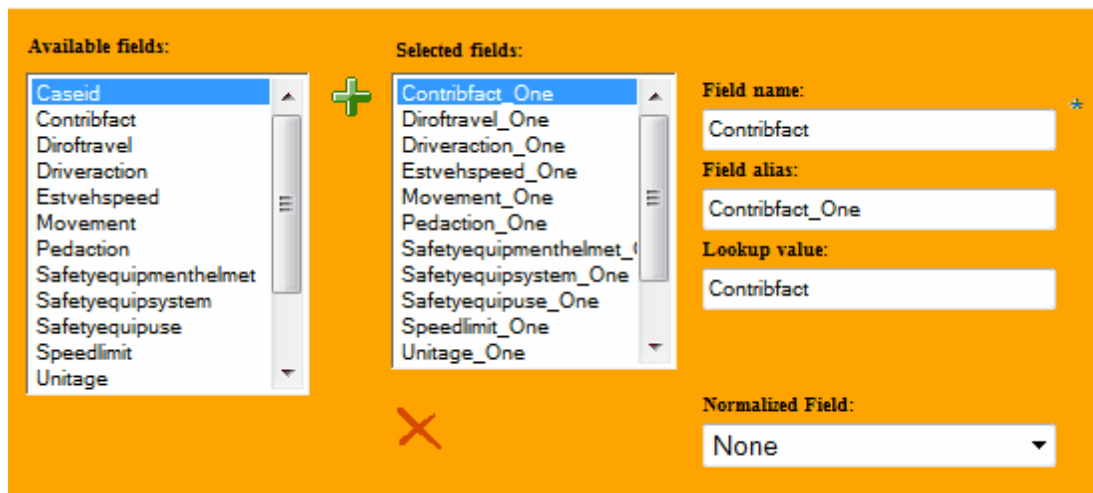
- Table drop down - Allows the user to move between the Selected joined tables
-  - Hides the table join in
- Table - Shows the name of the table selected
- Alias - Shows the alias name that will be used by the table in the query
- Join type - shows how the table will be joined in the from clause of the query(INNER, LEFT OUTER, RIGHT OUTER)
- Join conditions window - Shows the current join conditions for the table
-  - Opens the Add linkage condition box(The add linkage creates the join on clause of an SQL query)




The dialog box titled "Add linkage condition" has a yellow background. It contains the following fields: "Table Name: dbo.NODES", "Alias: Node", and "Join Type: LEFT OUTER". Below these is a dropdown menu currently showing "-- No Selection --" with a blue asterisk to its right. To the right of the dropdown is an equals sign "=" followed by an empty text box, also with a blue asterisk to its right. At the bottom are "OK" and "Cancel" buttons.

The drop down menu allows the user to select a field from the current table. The joined Table.Field can be entered in the box after the = sign.

-  - Delete the selected join condition
- Current selected field
- Current joined field
- Auto-suffix - Suffix added to any field alias in the currently selected table(Recommended alias are _ONE, _TWO, _THREE for the first three vehicles)






This interface is divided into three main sections. On the left, under "Available fields:", is a list box containing: Caseid, Contribfact, Dirotravel, Driveraction, Estvehspeed, Movement, Pedaction, Safetyequipmenthelmet, Safetyequipsystem, Safetyequipuse, Speedlimit, and Unitage. In the center, a green plus icon is positioned between the available and selected fields lists. Below it is a red X icon. On the right, under "Selected fields:", is a list box containing: Contribfact_One, Dirotravel_One, Driveraction_One, Estvehspeed_One, Movement_One, Pedaction_One, Safetyequipmenthelmet_One, Safetyequipsystem_One, Safetyequipuse_One, Speedlimit_One, and Unitage_One. To the right of the selected fields list are four input fields: "Field name:" with the value "Contribfact", "Field alias:" with the value "Contribfact_One", "Lookup value:" with the value "Contribfact", and "Normalized Field:" with a dropdown menu currently set to "None".

- Available fields - The list of fields from the current selected table
-  - Adds the selected field in the Available fields list to the Selected fields list
- Selected fields - These are the fields that will be returned by the query
- Field name - The name of the currently selected field in the Selected fields list
- Field alias - The field alias that will be used in the query (Any value from the Auto-Suffix box will be added to the field alias)
- Lookup value - This value must match the [DBField](#)^[275] name of the lookup query for the returned value to be looked up (Lookup values of !date, !time and !datetime can be specified for date, time and datetime fields respectively. These lookup values will cause the field to be formatted according to the locale information selected.)
- Normalized field - This drop down menu is used to flag the selected field as one of the [normalized fields](#)^[276] used by Crash Magic






WHERE:

The where clause section is used to add items to the where clause of the query. Where clause items should be avoided in study queries as the Crash Magic will dynamically where clause information.

-  - Add where clause (The editor will add an AND between items)
-  - Delete selected where clause
-  - Edit where clause

ORDER BY:

The order by section will add an order by clause to the query. Study queries must have an order by clause to ensure records will be returned in the same order. Users may experience difficulties in using collisions diagrams if the study order by clause is missing.

-  - Add an order by clause
-  - Delete selected order by clause
-  - Edit selected order by clause

System Panel

The system panel, located on the Administration side of the program contains several tabs:

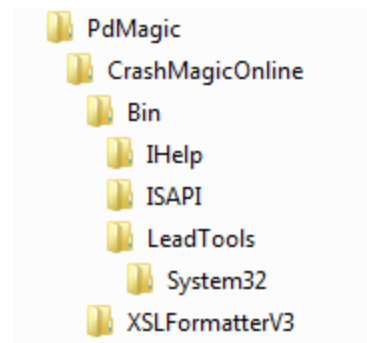
- **Server information**
This tab contains a list of variables and settings that define the current system configuration.
- **Current users**
This tab contains a list of the users currently logged into the program
- **Import users**
This tab provides the ability to import/initialize the list of users in the system based on a database query
- **Logs**
This tab provides access to the system stats and log files. Errors and general log information may be viewed online, and detailed debug information, if enabled, may be downloaded for additional analysis
- **Update**
This tab provides a list of optional updates to the system tables. Often these updates may be run directly from this tab. Sometimes, these items are merely instructional messages about required changes.

The program folders

When the program is installed, two folder trees are create/updated. One is in the "Program files" folder and the other is in the "Program data" folder. These folders are described here:

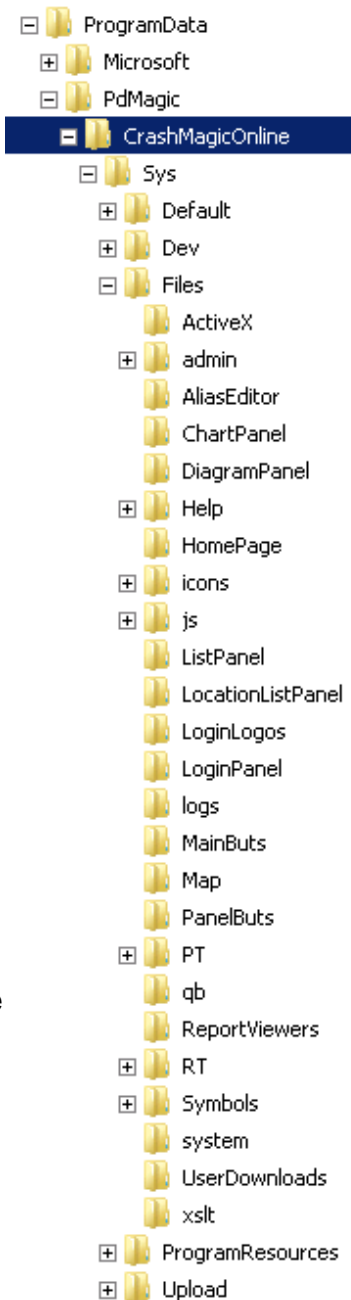
Program Files:

- **PdMagic** - Container for all Pd' Programming software applications installed on this computer.
- **PdMagic\CrashMagicOnline** - Container for Crash Magic folders.
- **CrashMagicOnline\Bin** - This folder contains the program files and dynamic libraries. (base2, etc, and hyphenation are components for the Antenna House PDF creator)
- **CrashMagicOnline\XSLFormatterV3** - This folder contains a programmer's toolkit used for creating PDF documents.
- **Bin\ISAPI** - This folder contains the ISAPI dll and configuration file used when installing Crash Magic as an IIS plugging.
- **Bin\LeadTools** - This folder contains a programmer's toolkit used for image manipulation.



ProgramData:

- **CrashMagicOnline\Sys** - Container for folders used by and managed by the system.
- **Sys\Default** - This folder contains runtime files created as the program operates.
- **Default\Cache** - This folder contains dynamic information prepared and removed as users access the system. Additional folders will be created below this one to support each currently logged in user as well as system temporary files.
- **Default\Logs** - the default folder for log files
- **Default\Templates** - Templates used for specific dynamic pages.
- **Sys\Dev** - This folder contains some sample code for developers using SOAP or MagicAuto calls to the system.
- **Sys\Files** - This folder contains static files needed by the program. For the most part, it contains graphics. It also contains javascript and some system templates.
- **Files\ActiveX** - Contains any needed ActiveX controls. Currently just contains the Silverlight diagram viewer XAP file.
- **Files\Icons** - these folders hold graphics used by the system. The following folders used to hold some of these graphics but are now deprecated: Admin, ChartPanel, DiagramPanel, ListPane, MainButs, PanelButs
- **Files\Help** - Holds files used when accessing the help system
- **Files\HomePage** - Holds some static html files that can be used for launching the program with window size and attributes pre-set.
- **Files\JS** - holds javascript used by the program.
- **Files>LoginLogos** - holds images of clients' agency logos for use on the login screen.
- **Files\ReportViewers** - Work in progress. Not used.
- **Files\RT** - holds temporary files created by the program, which will be served up to the client browser. This folder will be created and maintained by the program and will be regenerated automatically if removed.
- **Files\Symbols** - holds graphics that may be inserted into collision diagrams.
- **Files\System** - holds program error message files and templates.
- **Files\UserDownloads** - holds files that will be made available for download on the user's home page. By default, the application includes the Adobe SVG Viewer™.



- **Sys\ProgramResources** - Files used internally by the program. PdMasterShared is imported as needed at startup, connection strings are provided in connection dialogs as templates, ConfigStudyMap... is used to upgrade v2 user data to v3.
- **Sys\Upload** - Uploaded files section. For any user uploads, especially data import and conversion.

User

Each person who will access Crash Magic will need a user account. User accounts may not be shared. This is a technical limitation, not a licensing one.

Name: Enter you user name.

Description: Enter a description..

Login: Enter a login name. This is the User Id that will be entered at the Crash Magic login page. If each user has their own login to the collision database, then the login name entered in this field should match the login used for the database.

Password: Enter a password. This is the password that will be entered for the user at the login page.

View state: This should be set to Opened.

Email: Enter your email address. This will allow you to send PDF reports to other users.

Diagram Type: This drop down specifies how you would like to view your diagrams. The options are silverlight or svg. Pd' Programming recommends that this be left to silverlight.

Allow Analysis login: Check this box for the ability to log into the analysis section of Crash Magic.

Group Administrator: Check this box to grant the ability to login as the group admin. This Check box should only be checked for users that will be designated as the Group Admin. This allows users to change configuration items in the program.

Data Entry allowed: Check this box if the ability to use Crash Magic data entry is needed. This option will allow the user to download and use the data entry program.


Enable Cloneable logins: Allows the user to be copied for SOAP access. This should be left unchecked unless you are planning on accessing Crash Magic through a SOAP service.

Remote Access login allowed: Allows SOAP access of the user. This should be left unchecked unless you are planning on accessing Crash Magic through a SOAP service.

View SYS info: Allows the use to see system information(This feature is not currently enabled)

Enable Application Debugging: Allows the user to see debug information, and some features that have not been completed. This should be left unchecked.

Enable Help System Debugging: Allows the user to view system debug information. This should be left unchecked.

Now that a user with group admin privileges has been created Click the exit button to log out of Crash Magic as the Master Admin . The Master Admin login should no longer be used.

The group admin should be used for all reaming configuration tasks. To log in click on the show advanced options link of the Crash Magic login form. Enter the user Id and password just created then click the Group Admin button.

User Group

All user logins are members of a group. A single Crash Magic installation may have many user groups. A user group contains configuration information and attributes and templates used by all members of that group.

A special user group ".master" contains system-wide configuration and template information. Members of this group are considered "Master Admins" and may effect changes on any part of the Crash Magic system. Most administrators are just "Group Admins" which means that they are an admin for a specific user group only .

Only a Master Admin or Group Admin may access the Administrator side of the program. Group admins may make changes to their group only, but they may read and/or copy configuration and template objects from the .master group.

User group form

Crash Magic Administration - Windows Internet Explorer

Crash Magic Administration - Pd' Programming, Inc.

System
Master
.shared
CO@PdProgramming
.shared
scott

Summary Settings XML Data (advanced)

Name: CO@PdProgramming

Description:

Shared user: .shared

Auth domain:

Login Logo: PdProgramming.gif

BG Color: #FFFFFF

Encryption Method:

Encryption Key:

Utility Functions:

Delete all projects in this group

Ready Clipboard: Empty Scott (CO@PdProgramming) Wednesday, March 06, 2013 3:24 PM

Name - Change the name from User Group to your configuration name. All Crash Magic configuration names start with the State or Province name followed by an @ and the name of the Municipality or DOT with no spaces. Examples CO@Denver, ND@DOT CO@PdProgramming

Description - Set the description of the configuration. Example "City of Denver for crash report DR 2447"

Shared user - This user stores objects for the entire group. It can not be set until the [.shared user has been created](#)³³²

Auth domain - Specifies the domain that Crash Magic will use when [Active Directory](#)¹⁹⁶ is set under IIS.

Login Logo - Specify the file name of the login log that is displayed on the login page. This file can be a valid jpg or gif image placed in the <Crash Magic Program Data>\SysFiles\LoginLogos folder.

BG Color - Specify the hexadecimal background color for the login logo image.

Additional roadway systems

Additional roadway system such as highway or state defined road names...

Importing SYS configuration data

This article is specific to "on-site" installations where clients manage their own server. This article does not apply to "hosted" clients on PdMagic servers.

Crash Magic SYS data is all record types maintained by the system. This includes:

- User groups
- Users - Includes the .config and .shared users that hold the configuration resources
- PSRattrs - Also referred to as "resources" and "templates") These make up a client configuration.
- Projects
- Studies
- Reports

There are several methods available to manage SYS data:

- Manually, using the administration form on the program.
- Import, using the administration form on the program.
- Import, using the Login form's "Import configurations ..." option.

Crash Magic utilizes a "configuration" to interpret the crash data used by your agency. A configuration is simply a collection of resources, each referred to as a "PSRattr". Parts of every configuration are protected and editable only by PdMagic. Other parts of your configuration are available to be edited by you, the Crash Magic user. When using Crash Magic's hosted server, PdMagic maintains the protected part of your configuration for you. When Crash Magic is installed on-site, it is important that our clients are able to update the protected parts of the configuration as provided by PdMagic.

For some agencies, with a completely custom configuration, there is only the ".Master" configuration and the client configuration. However, in most cases there are several levels in play. You can view these levels by logging into the administration form in the program. There you will see a "tree" of user groups containing configurations. It will contain ".Master"; a state level configuration; often some intermediate, like county or state configurations; sometimes a PdMagic reports configuration; and then your agency configuration.



A typical configuration “tree” containing:

- **.Master group** – contains configuration resources common to all agencies and all users. This group of resources is updated with every version of the program and is not editable by any group administrators. Even PdMagic only updates this through the new version installation process. In fact, any changes to this configuration will be automatically overwritten with the next version. It’s essentially part of the program. This is considered a “protected” group.
- **“_” groups.** Groups that start with an underscore contain shared resources. Most agencies in a state will share the same database fields, lookups and overall configuration. They will all reference (inherit) the resources in these groups. An example group name might be “**_OR@CDS**” where the “OR” indicates the state and the “CDS” indicates the database or crash data system used by the agency. This type of group name might also be called “**_MD@Base**”. In this case, the “MD” is the state and “Base” indicates that this is the base configuration for all MD clients. Further down the list might be report resources created specifically for a client. For example “**_OR@CDS_Schematics**” are collision diagram schematics created specifically for the state of Oregon and only useful to clients with Oregon data. A group like this might also be called “**_MD@PdReports**”, again the first part names the state for which the reports work, and the second part “PdReports” describes the group’s contents. Most “_” groups are managed by PdMagic and are protected.
- **Agency group.** There will only be one agency group in any configuration. This group will almost always follow the naming convention of `<StateAbbreviation>@<AgencyName>`. (e.g. `OR@Eugene`; `MD@DOT`; `CO@Boulder`; etc.) This group contains content only useful to that agency. This group is also the only group that clients have access to edit. Clients may edit content in the .shared user or any of the named users. They can not edit the .config or the .tools user content. Those are provided by PdMagic and are protected.

When hosted on the PdMagic server, we update the protected portions of the configuration. Clients may use configuration inheritance to add / edit configuration resources in the .shared or individual user accounts. This makes it possible to customize and override all of the protected resources provided by PdMagic. However,

when hosted on an agency server it can be necessary to update the protected resources as PdMagic makes changes to them. These resources should be imported exactly as provided by PdMagic and not edited.

Imports of configuration resources (PSRattrs) are also known as SYS table imports. There are two types of configuration imports that an on-site administrator might need to perform. Both are the result of PdMagic making changes to your official configuration.

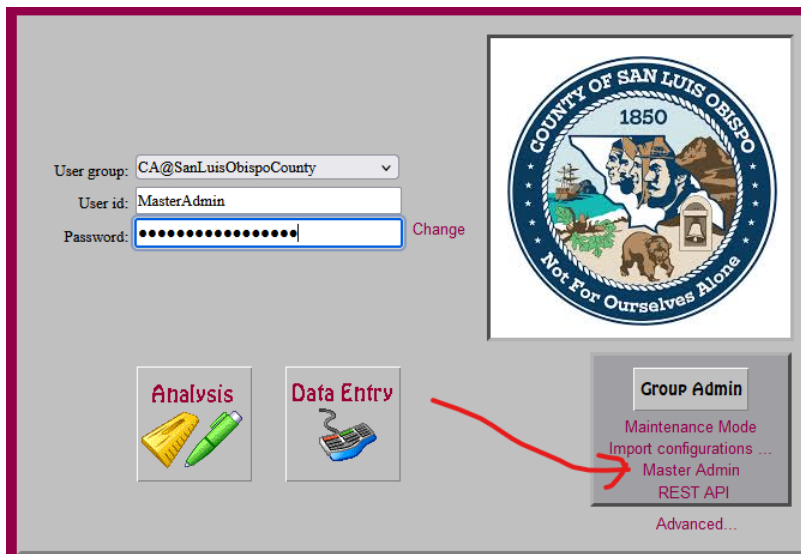
- Entire user group (shared resource). These are the “_” prefixed groups. These user groups are maintained as a unit containing a .config / .shared / .tools user. It is not appropriate to update just one or two resources within the group. Rather it is imperative to import the entire user group including all the user records in that group.
- .config within the agency group. Most of the resources in the agency group (<ST>@<Agency>) are not protected, but rather are owned by the group (.shared) or the individual user accounts. Only the .config user is protected.

These resources may be changed by logging in as the MasterAdmin. However, this is a precarious account to log into and makes possible changes that could completely disable the Crash Magic system. PdMagic is not responsible for repairing systems damaged by a client who logs in as a MasterAdmin. Instead a utility is available on the login form that can be used to update on-site shared user groups using an "Import configurations user".

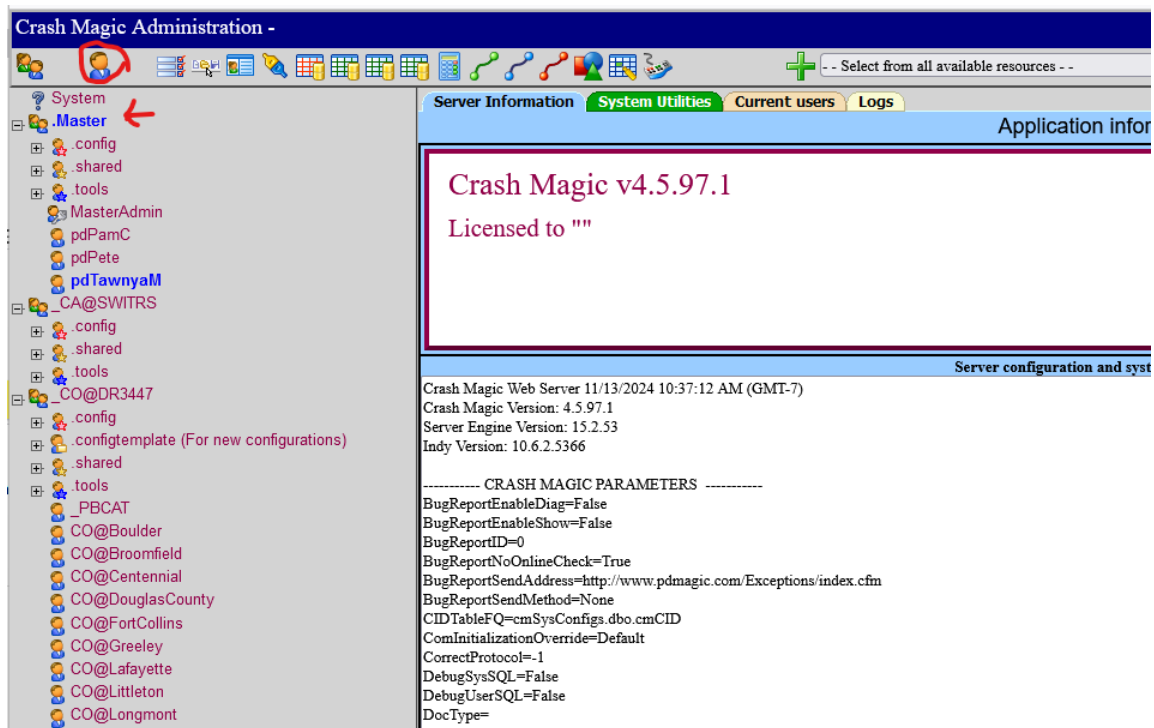
Create Import Configurations user

The import configurations option requires a special user account to access. Creating this user requires a one-time login as MasterAdmin. The list of people provided this login information should be extremely limited. Preferably to just to one person responsible for such upgrades.

Obtain the MasterAdmin password; Choose the advanced link on the login form and then click the "Master Admin" link. The user group selected is ignored for this type of login.



Be especially cautious while logged in as the MasterAdmin. Limit your activities to those that require this login and perform any other activities as a Group Admin if necessary. To create an Import Configurations user, select the .Master group in the tree on the left. Then click the New User icon in the button bar at the top.



This new user is very special. Note the red marks indicating necessary values. Provide this new user with:

- Name
- Login

- Password - It is strongly recommended that this password be different from a user's normal login password to prevent accidental logins to the wrong group.
- Email
- "Can Create User Groups" - this is an important one. It enables that user to perform imports from the login form.
- "Upgrade group list" - This is a comma-delimited list of the user groups that this user is permitted to import into from the login form. If this is empty, this user has no capabilities at all.
- No other permissions should be provided to this user.

The screenshot displays the 'Crash Magic Administration' web interface. The left sidebar shows a tree view of system resources, including 'Master', 'CA@SWITRS', 'CO@DR3447', 'IL@DOT', and 'PBCAT'. The main content area is titled 'Settings' and shows the configuration for a user named 'Example User'. The user's login is 'ExampleUser' and their email is 'Exampleuser@PdMagic.com'. The 'View state' is set to 'Opened'. The 'Base user settings' section includes a checked box for 'Can Create User Groups' and an 'Upgrade group list' field containing '_MD@Base_MD@PdReports_MD@'. The 'Basic permissions' section includes checkboxes for 'Allow Analysis login', 'Deny editing of PSRattrs', 'Allow Data Entry login', and 'Group Administrator'. The 'Special permissions' section includes checkboxes for 'Enable Remote Access login', 'Enable Cloneable logins', 'Must Be Cloned', 'Can Edit Group Config', 'Can use tools', 'Manage By Sync', and 'Can do surrogate login'. The 'Debug permissions' section includes checkboxes for 'Enable Application Debugging', 'Enable Help System Debugging', 'View SYS info', 'View SQL queries', and 'Hide progress reporting'. The 'DataSet caching' is set to 'Default'. At the bottom, there are buttons for 'Delete Selected PSRAttrs', 'Copy Selected PSRAttrs', and 'Promote to shared'. The status bar at the bottom indicates 'Ready - Wed 17:51:26 GMT'.

After adding this record, the ExampleUser may utilize the Login Form "Import configurations" option.

Import configurations

In order to import configurations from the Login Form. First enter the login and password for "Import Configurations" user. Now select the "Advanced" / "Import configurations..." link.

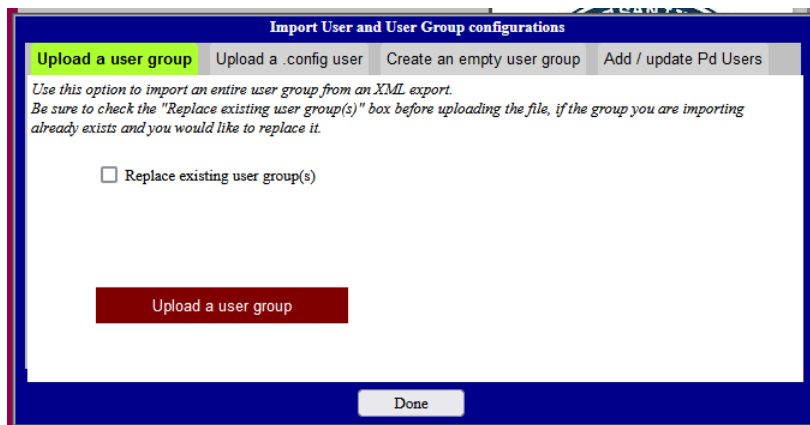
This dialog can be used to manage several types of SYS table resources.

- **Upload a user group** - This option is only available for shared user groups. These are user groups who's names start with an underscore. (e.g. _CO@Base) Use this option to import a single user group. While it can be used to import a new user group, this is rare. In most cases it will be used to overwrite an existing user group. Click on the "Replace existing user group" checkbox. You will be presented with a list of the groups that you are permitted to modify. Be sure that the group you are importing is in that list.

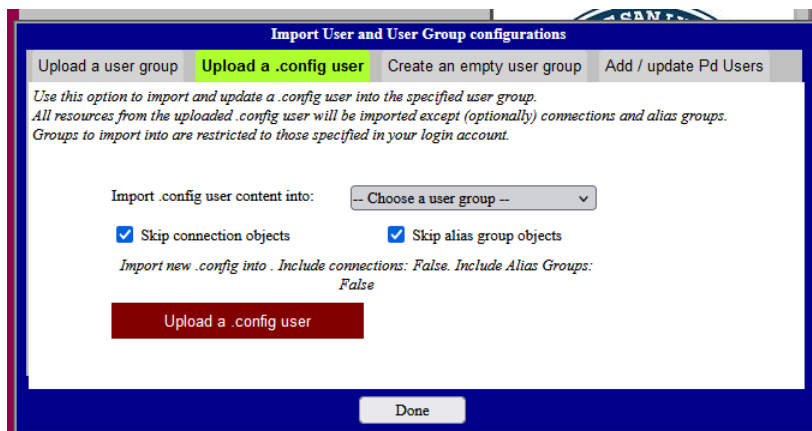
Important: This operation will start by deleting the existing user group. It will then proceed to import the new user group. Be sure this is your intent.

Click the "Upload a user group" button and select the FullUserGroup.xml file that you intend to import. Select okay. The import will proceed.

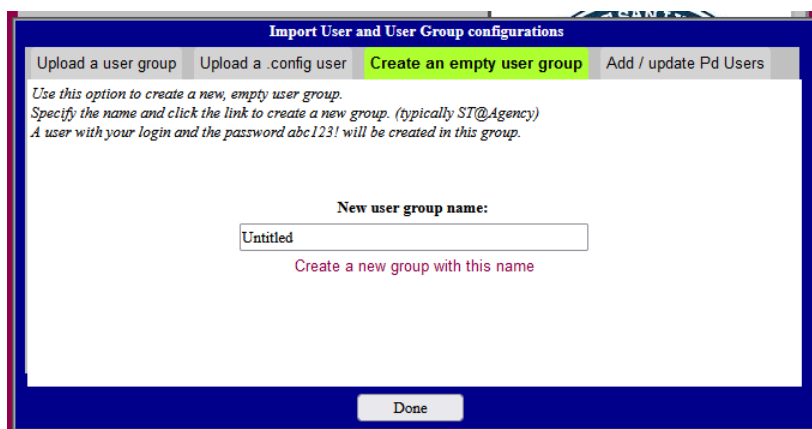
The new shared user group will now be available.



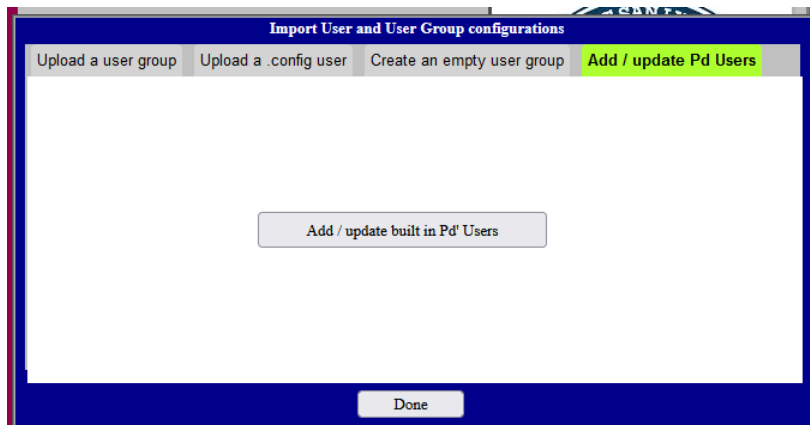
- **Upload a .config user** - This option is used to replace a .config user into a specified user group. Start by selecting the user group you intend to import the new .config user into. The upload file must contain exactly one user group. This will be used to validate against the selected user group. It must contain exactly one user named ".config".
Important: This process will start by removing all the content from the specified .config user and then importing the uploaded content. The two options titled "Skip connection objects" and "Skip alias group objects" are generally recommended to be checked. This preserves the existing connection and alias objects rather than replacing them with content from the import file. This is because the import file may have come from a different instance of Crash Magic that is connected to a different database. (e.g. DEV vs TEST) Checking this option preserves the existing connection settings for the installed instance. Aliases should generally not be replaced, because they are not stored with your configuration and the installed version may be more current than the what is in the import file.



- **Create an empty user group** - This option is only used when creating a brand new configuration. Usually this is performed by PdMagic. Specify the name of a user group that does not exist in the system. Then click the link to "Create a new group with this name". A new group will be created with a single user. (your login and the password shown in the dialog)



- **Add / update Pd Users** - This option takes no additional information. It simply iterates all the existing user groups and adds accounts for PdMagic support personnel. This option is generally only used by PdMagic.



5.5 Troubleshooting

Debug url parameters

When installing Crash Magic it is possible that conflicts, permissions, security, directory mappings or other other settings may prevent the program from displaying sufficient information to determine what is going wrong, In these cases, it can be helpful to be able to call the program in a manner that doesn't perform any database, file, or other actions with potential for failure. Available parameters are:

/DIAGnn

- 1 - Help only
- 2 - Request
- 3 - Event log test
- 4 - Log file test
- 5 - Exception test

The following parameters may be added to any DIAGnn command:

?DebugSQL=true|false - turns on or off sql logging for all users who log in from this point forward
 ?StatusLogLevel= Debug | Info | Important | Warning | Error | Exception | None - changes the log level for the application.

Note: In order for the server to start, it does perform some unavoidable database/file system access.prior to responding to this request.

Launch sequence and timing

The process of starting Crash Magic Online occurs differently for each of the application types. Here are those ways:

- Standalone - When lunched as a Windows dialog, (for debugging purposes) the application starts exactly when it is launched.

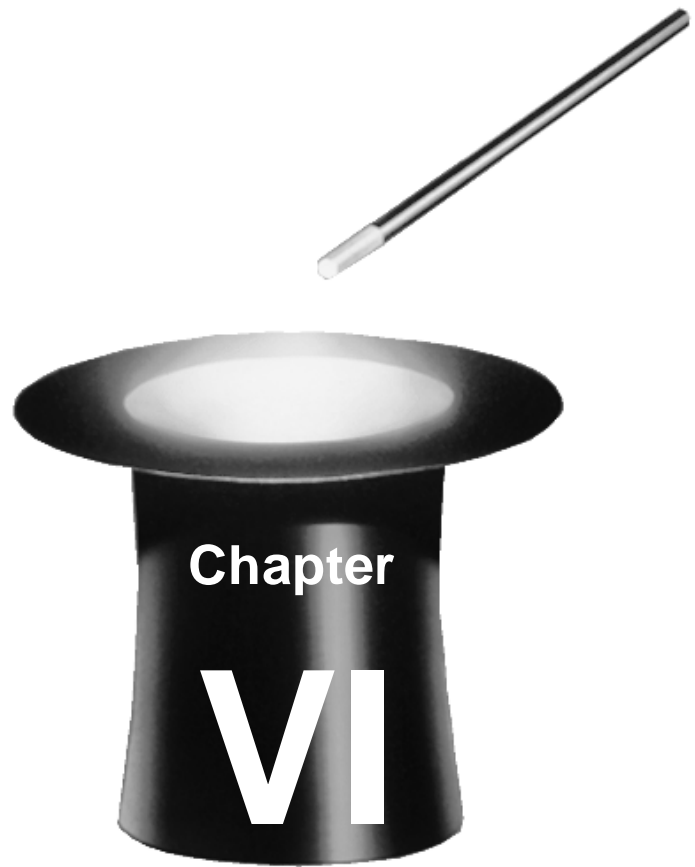
- Service - When installed as a Windows service, the application launches briefly to register itself, and is then idle until a request is made, at which point it launches again.
- ISAPI - When installed into IIS, any version, the program doesn't load until it's first request. It then remains loaded until its application pool is stopped.

Process log

Crash Magic uses several methods for logging status and error messages while running. These include:

- Windows Event Log
- Log files
- Log table in the cmSys database

At the time of this writing, only the XMLtoDB import process utilizes the Log table in the cmSys database. Most other log messages are written to the log files. The Windows Event Log is also used for those situations where the cmSys database can't be guaranteed to be available. (i.e. during system startup, shutdown and during errors related to writing to the system tables) Over time, all messages will be written to either the Windows Event Log or the cmSys database tables.



6 Utilities

6.1 PdForXML.exe

A command line utility that is designed to query a MS SQL Server database periodically and produce XML files representing the selected rows.

Help is available with the command

```
PdForXML --help
```

Top level commands

```
c:\>pdforxml --help
PdForXML 1.0.7222.23520
Copyright c 2018

all          Output all records to one file

byrecord     Output one record per file.

yearday      Output records grouped by Year and Day of Month.

examples     Display example usages.

help         Display more information on a specific command.

version      Display version information.
```

Three ways to write files

- **all** - all queried records are placed in a single file. For a small number of records, this can be very efficient and easier to manage than a directory full of xml files. However, if the result size is large or unknown, it can create large files that are more difficult to transfer, process and debug.
- **byrecord** - each file contains one record. This is the preferred method of transferring data to Crash Magic. Files are named using the specified caseid field plus optional prefix and suffix values.
- **yearday** - similar to "all", yearday places all files with the same year and day of the month into a single file. For example, all records with a datefield representing Jan 1, Feb 1, Mar 1... etc. for the year 2020 will be in the same file. This guarantees exactly 31 files per year. For most agencies, this is a reasonable number of records per file.

[all] parameters

```
c:\>pdforxml --help all
PdForXML 1.0.7222.23520
Copyright c 2018

--priortimeunit    Required. Unit (months,days,etc.) prior to current date to use.
                  Only days currently supported.
                  (Mutually exclusive of StartDate, EndDate)

--priortimevalue    Required. Number of units prior to current date to use. Current
```

	day will be added. (Mutually exclusive of StartDate, EndDate)
--startdate	Required. First date to process (inclusive). (Mutually exclusive of PriorTimeUnit, PriorTimeValue)
--enddate	Required. Last date to process (inclusive). (Mutually exclusive of PriorTimeUnit, PriorTimeValue)
--datefieldname	Required. Name of date field in primary table
--directory	Required. Output directory for XML files
--prefix	(optional) Prefix for output files
--suffix	(optional) Suffix for output files
--server	Required. MS SQL Server address
--database	Required. MS SQL Server database name
--login	Required. MS SQL Server login name
--password	Required. MS SQL Server login password
--queryfilename	Required. File containing main SQL query. This query requires the following to be in its WHERE clause: --{PdWhere}
--detaillog	(optional) Detailed log for debugging
--help	Display this help screen.
--version	Display version information.

[byrecord] additional parameters

--caseidqueryfilename	Required. File containing SQL query to get list of CaseIds. Case Id must be first column. This query requires the following to be in its WHERE clause: --{PdWhere}
--caseidfieldname	Required. Name of case id field in primary table

[yearday] additional parameters

none

PdForXML - Examples

The examples below cover a number of the extract scenarios you might run into. A common strategy is to use the first example on a regular basis, perhaps nightly, capturing the prior 30 days. Then less frequently, perhaps quarterly or semi-annually, query a larger range to capture corrections or late submissions.

1. ByRecord example: One record per file, previous 180 days.

```
PdForXML.exe byrecord --priortimeunit days --priortimevalue 180
--directory C:\OutDirectory
--server db.example.com\example --database EXAMPLE_DB --login Name --password Password
--caseidqueryfilename C:\Temp\caseidquery.sql
--queryfilename C:\Temp\examplequery.sql
--datefieldname CrashDateField --caseidfieldname c.CaseId
```

2. ByRecord example: One record per file, specify date range.

```
PdForXML.exe byrecord --startdate 2011/10/01 --enddate 2013/9/8
--directory C:\OutDirectory
--server db.example.com\example --database EXAMPLE_DB --login Name --password Password
--caseidqueryfilename C:\Temp\caseidquery.sql
--queryfilename C:\Temp\examplequery.sql
--datefieldname CrashDateField --caseidfieldname c.CaseId
```

3. All example: All records to one file, previous 180 days.

```
PdForXML.exe all --priortimeunit days --priortimevalue 180
--directory C:\OutDirectory
--server db.example.com\example --database EXAMPLE_DB --login Name --password Password
--queryfilename C:\Temp\examplequery.sql
--datefieldname CrashDateField
```

4. All example: All records to one file, specify date range.

```
PdForXML.exe all --startdate 2011/10/01 --enddate 2013/9/8
--directory C:\OutDirectory
--server db.example.com\example --database EXAMPLE_DB --login Name --password Password
--queryfilename C:\Temp\examplequery.sql
--datefieldname CrashDateField
```

5. YearDay example: Records grouped by Year and Day of Month, previous 180 days.

```
PdForXML.exe yearday --priortimeunit days --priortimevalue 180
--directory C:\OutDirectory
--server db.example.com\example --database EXAMPLE_DB --login Name --password Password
--queryfilename C:\Temp\examplequery.sql
--datefieldname CrashDateField
```

6. YearDay example: Records grouped by Year and Day of Month, specify date range.

```
PdForXML.exe yearday --startdate 2011/10/01 --enddate 2013/9/8
--directory C:\OutDirectory
--server db.example.com\example --database EXAMPLE_DB --login Name --password Password
--queryfilename C:\Temp\examplequery.sql
--datefieldname CrashDateField
```

Example SQL Query

```
SELECT
    c.CaseId as CaseId,
    c.Date as Date,
    'CO' as State,
    s1.StreetName as PrimaryStreet,
    s2.StreetName as CrossStreet
FROM Crash c
LEFT JOIN Streets s1 ON (s1.StreetId = c.PrimaryStreetId)
LEFT JOIN Streets s2 ON (s2.StreetId = c.CrossStreetId)
WHERE --{PdWhere}
```

```
ORDER BY PrimaryStreet, CrossStreet, c.CaseId  
FOR XML PATH( 'Crash' ), ROOT( 'Crashes' )
```

Example Case Id Query

```
SELECT CaseId  
FROM Crash  
WHERE --{PdWhere}
```

The query examples above each contain "--{Pdwhere}" in their WHERE clauses. This will be replaced with the appropriate date range during processing. If another WHERE clause needs to be included, prepare it such that PdWhere can still be inserted.

```
WHERE (Jurisdiction = 123) AND --{PdWhere}
```



7 Prepare a new configuration

7.1 Overview

A configuration is the group of attributes and templates (database records) that enable Crash Magic to read and analyze each different client's data. The configuration describes such basic things as the connection and query used to access the database; calculated fields that are used to "normalize" the data so that the program can understand it; street, intersection and node normalizers to provide location references; object maps to convert data to graphics; and many more pieces of information, all designed to enable the program to understand a new dataset.

In most cases, a configuration is purchased with the Crash Magic license. It is still appropriate to consider custom changes and tweaks to the provided configuration. This section of the manual describes the creation of a configuration from scratch. It is the process followed by Pd' Programming when we create configurations.

Preparing a configuration from scratch, the section which follows, can be complicated. Please read this section completely before beginning.

You will find that the first step in creating a configuration is to create a new user group. This requires that the software has been [installed](#)¹⁴⁷, and is accessible by browser. It also requires that you either have permissions to create a user group, or can log into the program as a Master Administrator to create the new user group.

Because of the powerful inheritance mechanism, a prudent method of making configuration changes is to copy the resources that will be changed into your own user account for editing. Once the object has been successfully tested under your own account, it can be promoted to the group for others to use. This also allows users to continue working without interruption until the new object has been completed. And, if mistakes are made, simply deleting the new object causes the program to go back to using the original resource. However, for creating a new configuration, it is usually best to build in in the .shared user, where it will eventually reside.

7.2 Resources

This section contains information that will be useful during the configuration.

At Pd' Programming, create appropriate folders in RawData, Configs and Docs for the client. Then use the pdClientManager to access:

- Rawdata - Arranged by country / state / agency. Each raw data extract / download is placed, as received, in a dated folder (e.g. U:\Rawdata\USA\CA\LakeForest\20180325_Switsr2017) Any manipulated versions of the data, even unzipping, reside in a subfolder named "pd".

- Configs - Also arranged by country / state / agency. (e.g. V: \Configs\CrashMagic\USA\CA\LakeForest\3_9\InProgress) See the ProgrammingGuidelines document for important versioning instructions.
- Configs metadata - stored at the agency level, or at the state level for statewide information such as forms, lookups, download details, etc.
- Docs projects - e.g. U: \docs\clients\USA\CA\LakeForest\Projects\20180325_Configuration should contain any information or backup work other than metadata. This folder will also house the ConfigurationVerification document.

Required and Recommended fields

Crash Magic was designed with the idea that collision data would not have to be modified to support the program, but some fields are required for analysis.

Required Data Structure:

Collision diagrams and studies in Crash Magic require a single record per collision. To accomplish this Crash Magic uses queries that flatten collision data from multiple tables into a single record for each collision. This means that a query must be able to extract collision information, vehicle one, vehicle two, vehicle three, pedestrian one, cyclist one and environmental information into a single crash record.

Data fields:

Crash Magic has no requirements for specific data types from collision data. It is possible to configure the program using text or numeric fields. However, any text fields must be consistent in order to produce reliable results. Non-validated, free-form data entry is not conducive to producing quality diagrams and reports. For example it can be impossible to determine a vehicle direction from an officer notes field. While large format fields are supported by Crash Magic(BLOBS and CLOBS), they should not be used for the required and recommended fields. The ideal structure is numeric or short text fields drawn from a predefined set of values like a lookup table

Fields that are absolutely required to generate a minimal diagram:

- Unique crash field name(s): A field or list of fields that can uniquely identify a crash across the entire set of data(Primary Key). This is used to be able to reference crashes within the diagram, as well as to provide feedback when "clicking on" a diagram graphic to obtain more information.
- Date: Date the collision occurred
- Location of crash (at least one of the following - all can be used simultaneously):
 - ✓ Primary and nearest cross street names: This is the most common means of identifying locations in urban and rural city settings. Jurisdiction information may be required for analysis on multiple locations with the same name. (For example jurisdiction information would be required for a database that contains information from the cities of Denver and Boulder, and both cities have an intersection location named Broadway and Arapahoe).

- ✓ Intersection node number: This must be a unique value that identifies a specific collision location.
- ✓ Route and Mile marker: Usually used in county or state DOT agencies
- ✓ Street and hundred block: Usually used in urban settings as a supplement to "Primary and nearest cross street names".
- ✓ XY coordinates: The XY location (Latitude and Longitude) for a collision.
- Vehicle sequence Crash Magic must be able to identify the first and second vehicles involved in a collision.
- Direction of travel for each of the involved vehicles.

Fields that are absolutely required to generate a minimal diagram:

- Type of collision: Is a field that indicates a collision was a sideswipe or rear-end. This is used when the first two vehicles are traveling in the same direction. Without it, all such crashes are rendered as rear-end collisions.
- Vehicle movements: for each of the first two vehicles involved in the collision. Values such as "left turn", "slowing", "backing", "straight", etc. are used to create the appropriate arrows for each graphic. Without this information, all vehicles will have to be represented with straight arrows.

Fields that are strongly recommended in order to properly annotate each collision graphic:

- Time of the collision: Users will not be able to use standard time charts without this field.
- Crash severity: Used to render an injury or fatality symbol associated with a collision graphic. This field should indicate at least "property damage"; "injury"; "fatality" across the entire crash. (i.e. all vehicles combined) A less convenient means of determining this is by examining each vehicle individually.
- Indication of pedestrian, bicycle or fixed object: Used to render an appropriate annotation to the graphic. If these values are represented as vehicle types, and given direction and movement, a more accurate diagram can be created with bicycle and pedestrian collisions. Otherwise a pedestrian or bike symbol is simply located in front of the first vehicle with no indication of direction or movement.
- Lighting condition: Used to render a "nighttime" symbol associated with the collision graphic.
- Driver or DUI condition(s): Used to a render DUI annotation on the crash, or for each appropriate vehicle.

Additional useful information:

- Distance and direction from the nearest cross street: These fields are used to place crashes in the appropriate positions in the diagram. Without it, many assumptions need to be made about vehicles in or near an intersection.
- Lookup tables or files: These tables/Files will be used to convert raw data values to user readable values.
- Collision images: Scanned images of police reports, collision photos, and collision location photos can be included within Crash Magic.

- Roadway section order: The creation of corridor diagrams schematics requires information to place roadway sections in their correct order. Street milepost studies order roadway sections by tenth of a mile sections based on the milepost number. Street address corridors order roadway sections by 100th block based on the block number.
- List of engineer-desired fields to be shown: Crash Magic diagrams and reports can be customized to display a default list of fields.

Required data for a configuration

There are several types of data required for creating a Crash Magic configuration. The first type is hard copy crash report forms from the data in use. For users that will be importing data into Crash Magic, a sample years worth of the data that will be imported is required, and a key to the files and fields. Clients that will be using an existing database will need to provide a copy of the database with a sample year. An existing data structure will also need documentation to identify fields for analysis. The following data items are required for creating a configuration.

Sample Hard Copy Crash Reports:

Hard copy crash reports are in several ways when creating a configuration. Their most important use is to compare hard copy reports with Crash Magic reports. As a final step in creating a Crash Magic configuration hard copy reports are compared the Crash Magic reports to ensure that collision diagrams are rendered correctly. Hard copy reports are also used when creating a data entry form. Data entry forms are laid out to follow the crash report. This allows a person entering data from a crash report to move with the crash form. In cases where a client does not have an existing data structure, the reports will provide possible values for Crash Magic fields that will be used by the configuration. Pd' Programming recommends at minimum of the following types of crash reports:

1. Overlay (if used) of crash report values
2. At least one injury or fatality crash
3. At least one single-vehicle crash
4. At least one multi-vehicle crash
5. At least one crash at an intersection
6. At least one mid-block / milepost crash - This is used to determine how distance and direction from intersection (or addresses) is coded
7. At least one left-turn crash
8. At least one rear-end crash
9. At least one pedestrian crash
10. At least one bicycle crash
11. Example reports of any data elements that you feel are unique or need extra attention when describing your data

A Years Worth of Sample Import Files:

Creating a configuration that will import data requires access to the sample data files that will be used for import. Crash Magic can be configured to import files. These files can be .xml or ASCII(.txt, .csv, .del). ASCII files must have a header row to describe the

field being imported. Clients with multiple files must provide information on how the files are linked together.

A Year of Data In a Crash database:

Clients providing a database must have at least a year of sample data to work with.

Configuration checklist

The resources required for a configuration can be created in almost any order. However, the steps and order provided here have been tested and result in the most efficient use of time and the highest likelihood of a complete configuration.

All items *shown in italics* represent creation of Crash Magic objects (mostly PSRattr)

Above red line:

√	Description	Name	Note
Create user group - ready to configure			
	<i>Create user group</i>	ST@City	
	<i>Create .shared user</i>	<i>.shared</i>	
	<i>Create resource: Configuration Notes</i>	<i>Default</i>	Document out-of-ordinary configuration choices and data here
Review and verify data			
	Review documentation for the source data		
	Review sample data		

Below red line:

√	Description	Name	Note
Create SQL scripts for data tables			
	<i>Create resource: SQLScript</i>	CreateCrashTables	Hopefully just one for all tables needed
	<i>Create resource: SQLScript</i>	InsertLookups	Inserts lookups. As many as needed to fit

			in 64k and for logical categories
	Create resource: SQLScript	DropCrashTables	Will be used only during configuration testing
	Instantiate database on DEV SQL Server		Permissions granted automatically using web interface
Create remaining basic configuration resources			
	Create resource: Locale Info	Default	non-US customers only
	Create resource: .options	Default	
	Locate and set agency logo	<ST>_<City>.png	250x250 pixels exactly. Must send to Scott or Pete to add to source.
Create and verify connections			
	Create resource: Connection	Reader	
	Create resource: Connection	Writer	Used for changing crash data, coordinate data, volume data, etc.
Create and test import definitions			
	Create resource: Lookup Query	Lookups	Prefer fewer queries to more. Performance penalty at login
	Create resource: ImportLookups	ImportLookups	Needed only if converting data (i.e. text to numbers) on import (rare)
	Create resource: DBtoXML	<Source>_DBtoXML	Needed if source data is not XML.
	Process DBtoXML		Use the resulting XML to create XMLtoDB
	Create resource: XMLtoDB	<Source_XMLtoDB	All configurations will need this unless 100% data entry
	Import using XMLtoDB		
Create analysis queries			

<i>Create resource:</i> Study query	Crashes	One per roadway system. (usually just one) Fill in normalize and lookup values
<i>Create resource:</i> Calculated fields	P_Default	All values required - basis for normalizing data
<i>Create resource:</i> Calculated fields	P2_Default	Currently used for GIS - required
<i>Create resource:</i> Calculated fields	U_UserSpecific	Handy for making object map clearer or providing custom fields for user
<i>Create resource:</i> query	ClickOn	
<i>Create resource:</i> query	ClickOnImages	For data that may have multiple images/columns per crash
Create normalizers		
<i>Create resource:</i> Streets query	Streets	For streets normalizer
<i>Create resource:</i> Intersections query	Intersections	For intersection normalizer
<i>Create resource:</i> Nodes query	Nodes	For node normalizer. If nodes used
<i>Create resource:</i> Street normalizer	Default	"Default" is for the primary road system
<i>Create resource:</i> Intersection normalizer	Default	
<i>Create resource:</i> Node normalizer	Default	
Create study definitions		
<i>Create resource:</i> study definition	Date range	
<i>Create resource:</i> study definition	Case id	
<i>Create resource:</i> study definition	Intersection...	Intersection, Route milepost, Address, UserCID, XY Rect, etc.
Create object map		
<i>Create resource:</i> Object map	Default	
Add veh 1 and veh 2 movements		
Add veh 3 direction symbol		

	Add bicycle, pedestrian, direction and no direction		
	Add injury, fatality, DUI, nighttime symbols		
	Add any remaining attributes		
Create templates			
	Create resource: Field list	ClickOn	from analysis side
	Create resource: Field list	ClickOnImages	from analysis side
Prepare data entry			
	Create resource: Data entry definition	LongForm, ShortForm, etc.	
	Create resource: Validation rules	Default	
Misc. items			
	Create resource: .Options report buttons	Default	User specific templates
Complete verification			
	Prepare verification document		
	Have verification document reviewed		

Tools and utilities

- Notepad++
 - The configuration process will involve editing large text files, XML and other file formats. This may be performed with any capable text editor. Pd' Programming recommends Notepad++ for this purpose. It is good at handling large files, and provides formatting, syntax highlighting and data validation. Notepad++ can be downloaded, free of charge, from <https://notepad-plus-plus.org/>

The following Notepad++ Plugins have been useful when preparing configurations at Pd' Programming:

- XML Tools - Provides XML validation and formatting. Also provides XPath evaluation
- Compare - Compares two text files side-by-side.

7.3 Obtain data and metadata

Obtain and verify the following:

1. A crash report form for the state
2. List of fields and data types
3. Lookup fields
 - a. Make sure lookup values are listed for every lookup field
4. Determine how the tables are linked
5. Get validation rules from the client
6. Determine number of roadway systems (local vs highway)
7. Determine which study definitions are needed
8. Bring the file into Excel or other SQL viewer
 - a. Create or use a [schema.ini](#)²⁸⁶ if necessary
9. Browse field values
 - a. Make sure types are coming across correctly
10. Compare lookup tables with fields
11. Check that vehicle type includes Pedestrian and Pedcycle
12. Determine ClickOn GIS
13. At Pd' Programming: If importing from IMW database, Check for * in the street names if applicable. This would indicate potentially corrupt data from IMW v6.60 bug

Check data against [Required and Recommended](#)³²³ fields

User questions

Ask user questions (only one time)

- Inform client an effort will be made to add lookups for yes no fields, but fields that do not have lookups will only display the raw text values.

Wait for user answers

- Once the questions have been answered – return to [Obtain data and metadata](#)³³⁰ to review

Document any anomalies in Configuration Notes

Prepare to start the configuration

This is the last step before beginning the configuration process.

To avoid encountering common problems later on that require a complete restart, be sure that all prior steps have been completed.

Do not proceed to configuration unless all resources are in place and validated

7.4 Create and Prepare Users and Groups

This section contains the steps to creating a configuration. It is based on the Configuration Checklist in the prior section. The steps described here are high-level, sometimes containing tips or suggestions, and often standardized names or other information. However, each of the steps also refers to one or more reference topics that fully describe the details of the resources, forms or other tools needed to fully understand the step.

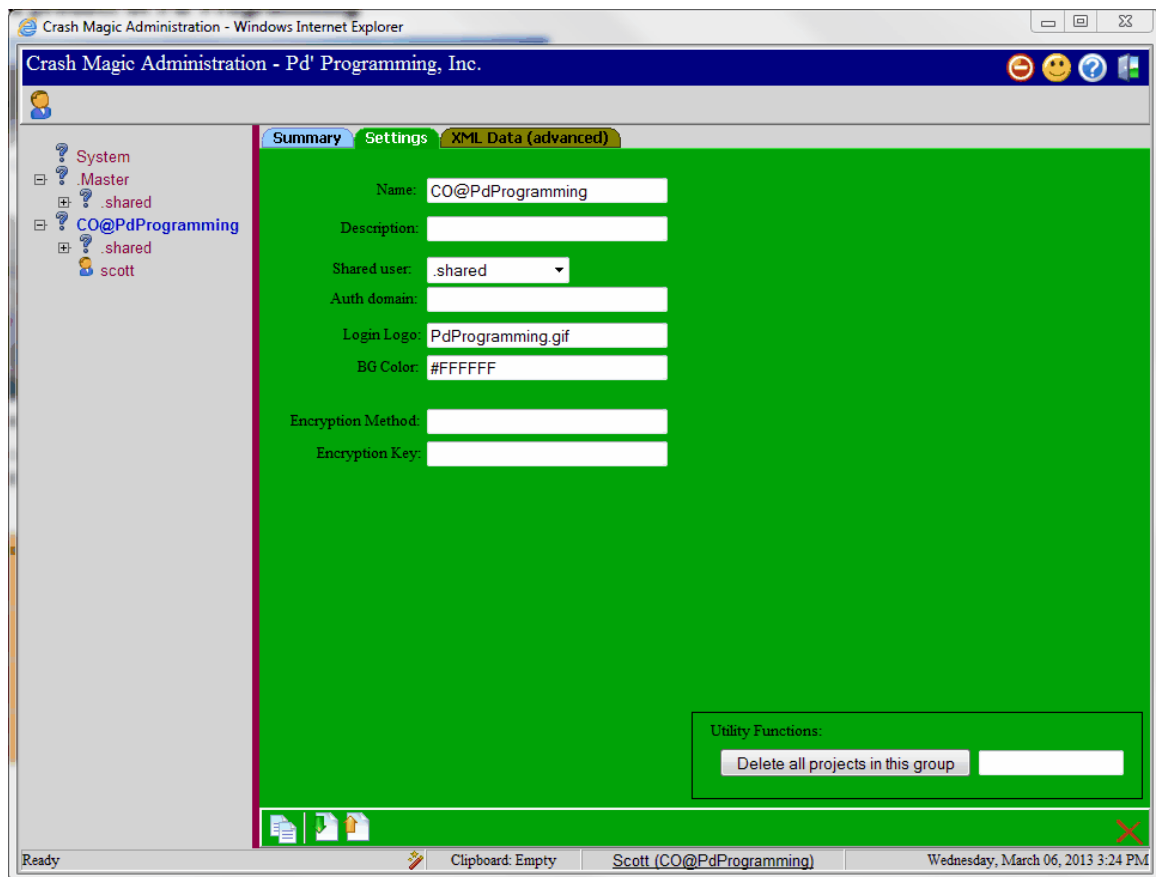
Create the User Group

In order to create a User Group, you must have permission to do so. Two types of users have this permission. Master Admin can create User Groups, and any user in the .Master group that has the "Can Create User Groups" permission. Only a Master administrator can give you either of those permissions.



Login screen with "Show advanced options" selected

The login screen contains a link for "Create User Group". Enter a login name and password that has permissions to create a new group and select that link. A new group called "untitled" will be created, along with instructions for logging into that group. Login to GroupAdmin using that information.



Select your new untitled group and rename it. The convention used by Pd' Programming is ST@Agency where "ST" is the state abbreviation, and agency is the agency name. (i.e. CO@DOT; CO@Boulder; CA@SanLouisOpisbo; LA@DOTD)

Create and update users

1. Select the newly created user (your login name)
2. Set an appropriate password
3. Select the "Allow Analysis login" and "Group Administrator" checkboxes

For Pd' supported configurations - add admin user logins for Pd' configuration engineers as well

Create the .config User

Create a .config user by selecting the "Create new user" icon at the top of the screen 

- Enter ".config user" as the name
- Enter ".config" as the login

- Leave the password blank. This is not an account that will be logged into.

Create the .shared User

Create a .shared user by selecting the "Create new user" icon at the top of the screen



- Enter ".shared user" as the name
- Enter ".shared" as the login
- Leave the password blank. This is not an account that will be logged into.

Log out. Log in.

In order to register these newly created objects, it is necessary to log in again.

1. Exit Crash Magic
2. Enter your log in information
3. Select the new user group

7.5 Create common PSRattrs

Most configurations need the same group of PSRattrs to get started. This section describes each one.

.Options - General

- Default
 - Fill in the appropriate values.

Raster image

- LoginLogo
 - Locate an agency logo on the web.
 - Format to exactly 250 x 250 pixels.
 - Save as PNG.
 - Upload that image.

Connection

- Reader
 - Use your own DB login and password.
 - Set the connection string to connect to the crash database.
 - If needed, the database will be created and tested in a later step.

- Writer
 - To be used for importing and/or data entry. This PSRattr is not needed if connecting directly to a private DB
 - Use the same connection string as for the "Reader".
 - Use your own DB Admin login and password.

SQL script

- CREATE crash tables
 - CREATE statement for tables using source/documented field types if possible, otherwise select field types by reviewing content. Preference to integer types for categorical fields when possible. Scripts should include primary key constraint and appropriate indexes based on Query - Study content. Typical tables will be:
 - Crash
 - Vehicle
 - Party
 - Streets (can usually be copied from another configuration)
 - Nodes (only if using a node field - rare)
- DELETE from crash tables
 - DELETE statement for crash data. Leave table structure intact.
- DROP crash tables
 - DROP statement for each crash data table
- CREATE lookup tables
 - CREATE statement for lookup table.
 - Fields must match columns in [Query - Lookups](#)²⁷⁵.
 - Can usually be copied from another configuration.
- INSERT lookups
 - Multiple INSERT statements to populate the lookup table(s).
- DROP" lookup tables
 - DROP statement for each lookup table

Text lines

- Schema.ini
 - Only needed if source data is CSV or ASCII fixed-field.
 - Will be referenced by [DBtoXML](#)²⁵³.

Import - DB to XML

- <BasedOnSource>
 - Name should describe source, as above.

- If no DB to XML PSRattr will exist, (source data is XML) a placeholder should be created. It should be named the same as this PSRattr with "Placeholder" as the suffix.

Import - XML to DB

- <BasedOnSource>
 - Only needed if source data is not XML.
 - Name should describe source. (e.g. "PD DB to XML" or "IMW DB to XML" or "SWITRS DB to XML")
 - This [DB to XML](#)^[253] PSRattr is essentially one or more queries against the raw data source.
 - The resulting query results are structured as XML and saved to files by the Convert process.
 - Can use [Configuration Helper](#)^[224] to create a simple version

Locale info

- Default
 - Only needed if not a US company or agency.
 - Enables different date/time formatting options.

7.6 Create and populate crash database

In order to proceed at this point, a database will need to be created and crash data imported. If connecting to an existing crash database with compatible lookup tables, this step may be skipped.

Create database and tables

1. Create a database to house the crash data for this configuration. Identify a login account that has read-only access to this data. If data entry will be performed, also identify an account with read-write access to this database. Important: The account with read-write access should not have DDL permissions.

At Pd' Programming: Use the tool at inhouse.pdmagic.com/db/ to create a database in "Configs". In addition to creating the DB in the appropriate place based on its use, it will also create appropriate logins and permissions.

2. Locate the Connection / Reader created [earlier](#)^[333], click the "Test" button. Once that works, do the same for the Writer if there is one.
3. Execute the CREATE crash tables script from the Execute tab.
Note: When executing an SQL script, the connection logins are not used, just the connection string. This is because SQL scripts can easily delete or restructure data. As a result, a login that has DDL permissions is required to execute the script. Select the "Reader" connection and enter a login and password with DDL permissions.
4. Execute the CREATE lookup tables script

Prepare a sample source file

When preparing a configuration it is best to have access to a sample of at least 2 years of crash data. It is best to have all the data, but 2 years generally makes a reliable configuration possible.

While preparing a configuration, it will often take several attempts to be successful at converting and importing the source data. For this reason, it is best to create a smaller extract of the full sample. This will allow for repeated convert / import tests with the least time spent waiting for the importer.

If using...

- Individual XML files, this is simply a matter of zipping a few together into a test zip file.
- A single large XML file, use notepad++ to extract a dozen or so crashes from the larger file into a single smaller one.
- A database query, simply reduce the query to a single or several days worth of data
- CSV or other files that require converting, there are two options:
 - Edit the CSV file to only contain a dozen or so records. This can be a little tricky if there are several related files.
 - Plan to use the [Import - DB to XML](#)²⁵³ queries to limit the converted data with a WHERE clause. After completing the initial import process, remove the WHERE clause to allow for all records to be processed.

Convert source data to XML

Use the [Import - DB to XML](#)²⁵³ to convert the source data files to XML. This PSRattr type does not have a proper editor at this time. Creation and editing is performed directly in XML within the program.

Once the DB to XML PSRattr has been completed, it can be tested from the convert tab on the Admin / User Group panel.

7.7 Configuring Corridor Diagrams

The first step in creating a corridor diagram is to create a query that can select the desired crashes.

Corridor diagrams require each crash record to contain a value to calculate the location of a crash within the corridor.

temporary

Preparing a Query-based corridor schematic:

A Query that returns one row per schematic (road section) required.

This query should include enough information to calculate the schematic name, section label, section condition. Field names do not matter, as calculated fields will be used to extract the data.

A corridor schematic

A corridor schematic contains a section `Schematic/DynamicContent/` that contains a list of `ContentSpec` tags. Each `ContentSpec` tag contains a `<Query>` tag indicating the query that should be used (if this is a query-based dynamic schematic) A `SequenceDirection` tag indicating the direction that his corridor travels towards. A list of parameters, (if this is a query-based dynamic schematic) The parameters are values that will be passed to the query. The Name indicates the parameter name, the Value should either be static, or start with a tilde (~) to cause it to be parsed. For example, `~Study.QueryParam.ASCII("", "", "AParam")` will return the current study's SQL parameter called `AParam`. The `DataType` specifies the data type of the parameter (i.e. Number, String, Date, etc.)

A calculated fields record containing three fields. (`AreaCondition`, `AreaLabel` and `Schematic`)

`AreaCondition` should resolve to a string that can be parsed to an expression and returns true if a crash belongs in that schematic/road section.

`AreaLabel` should resolve to a string that is the label placed next to that schematic section.

`Schematic` should resolve to a string that is the name of the schematic to use for the current section.

Preparing a sequence-based corridor schematic:

A corridor schematic

A corridor schematic contains a section `Schematic/DynamicContent/` that contains a list of `ContentSpec` tags. Each `ContentSpec` tag contains a `<First>`, `<Last>` and `<Increment>` tag. First and Last describe the start and end of the road section to be rendered. The values in the tags may be static (probably useless) or may start with a tilde (~) which indicates it should be parsed before use. For example, `~AsNumber(Study.QueryParam.ASCII("", "", "FirstMilepost"))` will return the value used in the current study for `FirstMilepost`. `Increment` tells how much to increment the range by with each iteration. This value should generally point to a program variable, but may be static as well. A `SequenceDirection` tag indicating the direction that his corridor travels towards.

7.8 Configuring High Crash Location reports

Requirements for High Crash Location Lists

High Crash Location reports are currently only valid for databases that locate their crashes based on street and cross street. For a database to be able to use High Crash Location reports crash records must be able to link to the following items:

- Primary Street Field
- Cross Street Field
- Date Field

Additionally, these streets must be able to be located to a specific city as street names can repeat across states.

Required Crash Magic Attributes:

- Distinct Streets query - This query returns a data set that contains a list of unique streets in the database. It should contain the street name, along with unique 1 and unique 2 if needed.
- Distinct Intersection query - This query returns a data set that contains a list of unique locations in the database. It should contain the primary street name, cross street name, and unique fields as needed.
- Street Normalizer - This PSRAAttr defines what a street is.
- Intersection Normalizer - This PRAAttr defines what an intersection is.
- All Data query that has the same fields as specified in the Distinct Intersection query.
- Node button enabled in the button list.
- xsl-locationlist - A XSL style sheet to display the results.(This should be located in your Master .shared user)

Step By Step

1. Create a distinct streets query. This query should return all of the streets in the database. It will usually be a distinct query that returns the primary street, with a union to the same query that returns all cross streets.
2. Create a new [normalize-streets](#)²⁶⁶ attribute.
3. Create a distinct intersections query. This query should return all of the intersections in the database. It will usually be a distinct query that returns that primary street, cross street, and unique fields.
4. Create a new [normalize-intersections](#)²⁶⁴ attribute.
5. Create a query based on the default query that contains :FirstDate and :LastDate as parameters. Street field names should match the intersection query.
6. Inside of the .options enable the alldatastudy and locationlist buttons.

Distinct intersection query

The Distinct Intersection query is used to collect all of the distinct intersections from with in an entire database. Since most systems do not already have a table of intersections like this, we query the crashes database for the distinct primary and cross street values.

In this case, it is also important to include a UNION statement to assure that the list includes both possible intersection names: Primary & Cross as well as Cross & Primary. Note the highlighted section where the street names are reversed. Changing the alias names does not accomplish this "flip flop".

As of this manual printing, the only way to accomplish a UNION such as this is to directly edit the XML for the Query:

1. Make sure there is no content between the <UnionQueries>. If there is, remove it.
2. Copy the entire content of the primary <QueryDefinition>
3. Paste the content inside the <UnionQueries> tags of the primary QueryDefinition. This is the "Second Query Definition"
4. In the Second Query Definition, set the <UnionType> to UNION.
5. In the Second Query Definition, Identify the <JoinPair> that joins the primary and cross streets. Reverse them as shown in the examples.

Example with 2 unique qualifiers

Example from Oregon DOT. In this example each street's uniqueness is dependent on both a Cnty_Id and City_Sect value. Note though, that in this particular case, the source data does not include Cnty_id and City_Sect_id for the streets, just for the crash.

As a result, it is not possible to identify all crashes at intersections on county or city boundaries unless the crashes are coded consistently to a single agency. Also, this example sorts by county, city, street and cross street. This is not required, but makes examination of data query results easier.

```
SELECT
Crash.Cnty_Id AS Cnty_Id,
Crash.City_Sect_Id AS City_Sect_Id,
S1.StreetName AS St_Full_Nm,
S2.StreetName AS Isect_St_Full_Nm

FROM
( dbo.CRASH Crash
LEFT JOIN dbo.Streets S1 ON ((S1.StreetNum=Crash.St_Full_Id) AND (S1.CNTY_ID=Crash.Cnty_Id) AND
LEFT JOIN dbo.Streets S2 ON ((S2.StreetNum=Crash.Isect_St_Full_Id) AND (S2.CNTY_ID=Crash.Cnty_Id) AND

UNION

SELECT
Crash.Cnty_Id AS Cnty_Id,
Crash.City_Sect_Id AS City_Sect_Id,
S1.StreetName AS St_Full_Nm,
S2.StreetName AS Isect_St_Full_Nm

FROM
( dbo.CRASH Crash
LEFT JOIN dbo.Streets S1 ON ((S1.StreetNum=Crash.Isect_St_Full_Id) AND (S1.CNTY_ID=Crash.Cnty_Id) AND
LEFT JOIN dbo.Streets S2 ON ((S2.StreetNum=Crash.St_Full_Id) AND (S2.CNTY_ID=Crash.Cnty_Id) AND

ORDER BY
Cnty_Id, City_Sect_id, St_Full_Nm, Isect_St_Full_Nm
```

Raw XML. The second (copied) query definition is highlighted in blue. The changed union type and streets are highlighted in yellow as they are above.

```
<pdroot>
  <Settings/>
  <QueryDefinition>
    <ExprQualifier>C</ExprQualifier>
    <UnionType/>
    <ConnectionName>Reader</ConnectionName>
    <LinkageList>
      <Linkage>
        <Table>dbo.CRASH</Table>
        <Alias>Crash</Alias>
        <JoinType>INNER</JoinType>
        <DefaultFieldSuffix/>
        <FieldList>
          <Field>
            <Name>Cnty_Id</Name>
            <Alias>Cnty_Id</Alias>
            <LookupKey>CNTY_ID</LookupKey>
            <MimeType>text/plain</MimeType>
            <Enabled>True</Enabled>
            <NormalizedField>None</NormalizedField>
            <DataType>FixedChar</DataType>
          </Field>
          <Field>
            <Name>City_Sect_Id</Name>
            <Alias>City_Sect_Id</Alias>
            <LookupKey>CITY_SECT_ID</LookupKey>
            <MimeType>text/plain</MimeType>
            <Enabled>True</Enabled>
            <NormalizedField>None</NormalizedField>
            <DataType>Integer</DataType>
          </Field>
        </FieldList>
        <JoinPairList/>
      </Linkage>
      <Linkage>
        <Table>dbo.Streets</Table>
        <Alias>Sl</Alias>
        <JoinType>LEFT OUTER</JoinType>
        <DefaultFieldSuffix/>
        <FieldList>
          <Field>
            <Name>StreetName</Name>
            <Alias>St_Full_Nm</Alias>
            <LookupKey>None</LookupKey>
            <MimeType>text/plain</MimeType>
            <Enabled>True</Enabled>
            <NormalizedField>None</NormalizedField>
            <DataType>WideString</DataType>
          </Field>
        </FieldList>
        <JoinPairList>
          <JoinPair>
            <Field>StreetNum</Field>
            <Value>Crash.St_Full_Id</Value>
          </JoinPair>
        </JoinPair>
      </Linkage>
    </LinkageList>
  </QueryDefinition>
</pdroot>
```



```

        <Field>CNTY_ID</Field>
        <Value>Crash.Cnty_Id</Value>
    </JoinPair>
    <JoinPair>
        <Field>CITY_SECT_ID</Field>
        <Value>Crash.City_Sect_Id</Value>
    </JoinPair>
</JoinPairList>
</Linkage>
<Linkage>
    <Table>dbo.Streets</Table>
    <Alias>S2</Alias>
    <JoinType>LEFT OUTER</JoinType>
    <DefaultFieldSuffix/>
    <FieldList>
        <Field>
            <Name>StreetName</Name>
            <Alias>Isect_St_Full_Nm</Alias>
            <LookupKey>None</LookupKey>
            <MimeType>text/plain</MimeType>
            <Enabled>True</Enabled>
            <NormalizedField>None</NormalizedField>
            <DataType>WideString</DataType>
        </Field>
    </FieldList>
    <JoinPairList>
        <JoinPair>
            <Field>StreetNum</Field>
            <Value>Crash.Isect_St_Full_Id</Value>
        </JoinPair>
        <JoinPair>
            <Field>CNTY_ID</Field>
            <Value>Crash.Cnty_Id</Value>
        </JoinPair>
        <JoinPair>
            <Field>CITY_SECT_ID</Field>
            <Value>Crash.City_Sect_Id</Value>
        </JoinPair>
    </JoinPairList>
</Linkage>
</LinkageList>
<WhereClauseList>
</WhereClauseList>
<OrderByList>
    <OrderBy>
        <Field>Cnty_Id</Field>
        <AscDesc>Asc</AscDesc>
    </OrderBy>
    <OrderBy>
        <Field>City_Sect_Id</Field>
        <AscDesc>Asc</AscDesc>
    </OrderBy>
    <OrderBy>
        <Field>St_Full_Nm</Field>
        <AscDesc>Asc</AscDesc>
    </OrderBy>
    <OrderBy>
        <Field>Isect_St_Full_Nm</Field>
        <AscDesc>Asc</AscDesc>
    </OrderBy>
</OrderByList>

```

```

    <UnionQueries>
      <QueryDefinition>
        <ExprQualifier>C</ExprQualifier>
        <UnionType>UNION</UnionType>
        <ConnectionName>Reader</ConnectionName>
        <LinkageList>
          <Linkage>
            <Table>dbo.CRASH</Table>
            <Alias>Crash</Alias>
            <JoinType>INNER</JoinType>
            <DefaultFieldSuffix/>
            <FieldList>
              <Field>
                <Name>Cnty_Id</Name>
                <Alias>Cnty_Id</Alias>
                <LookupKey>CNTY_ID</LookupKey>
                <MimeType>text/plain</MimeType>
                <Enabled>True</Enabled>
                <NormalizedField>None</NormalizedField>
                <DataType>FixedChar</DataType>
              </Field>
              <Field>
                <Name>City_Sect_Id</Name>
                <Alias>City_Sect_Id</Alias>
                <LookupKey>CITY_SECT_ID</LookupKey>
                <MimeType>text/plain</MimeType>
                <Enabled>True</Enabled>
                <NormalizedField>None</NormalizedField>
                <DataType>Integer</DataType>
              </Field>
            </FieldList>
            <JoinPairList/>
          </Linkage>
          <Linkage>
            <Table>dbo.Streets</Table>
            <Alias>Sl</Alias>
            <JoinType>LEFT OUTER</JoinType>
            <DefaultFieldSuffix/>
            <FieldList>
              <Field>
                <Name>StreetName</Name>
                <Alias>St_Full_Nm</Alias>
                <LookupKey>None</LookupKey>
                <MimeType>text/plain</MimeType>
                <Enabled>True</Enabled>
                <NormalizedField>None</NormalizedField>
                <DataType>WideString</DataType>
              </Field>
            </FieldList>
            <JoinPairList>
              <JoinPair>
                <Field>StreetNum</Field>
                <Value>Crash.Isect_St_Full_Id</Value>
              </JoinPair>
              <JoinPair>
                <Field>CNTY_ID</Field>
                <Value>Crash.Cnty_Id</Value>
              </JoinPair>
              <JoinPair>
                <Field>CITY_SECT_ID</Field>
                <Value>Crash.City_Sect_Id</Value>

```

```

        </JoinPair>
    </JoinPairList>
</Linkage>
<Linkage>
    <Table>dbo.Streets</Table>
    <Alias>S2</Alias>
    <JoinType>LEFT OUTER</JoinType>
    <DefaultFieldSuffix/>
    <FieldList>
        <Field>
            <Name>StreetName</Name>
            <Alias>Isect_St_Full_Nm</Alias>
            <LookupKey>None</LookupKey>
            <MimeType>text/plain</MimeType>
            <Enabled>True</Enabled>
            <NormalizedField>None</NormalizedField>
            <DataType>WideString</DataType>
        </Field>
    </FieldList>
    <JoinPairList>
        <JoinPair>
            <Field>StreetNum</Field>
            <Value>Crash.St_Full_Id</Value>
        </JoinPair>
        <JoinPair>
            <Field>CNTY_ID</Field>
            <Value>Crash.Cnty_Id</Value>
        </JoinPair>
        <JoinPair>
            <Field>CITY_SECT_ID</Field>
            <Value>Crash.City_Sect_Id</Value>
        </JoinPair>
    </JoinPairList>
</Linkage>
</LinkageList>
<WhereClauseList>
</WhereClauseList>
<OrderByList>
    <OrderBy>
        <Field>S1.StreetName</Field>
        <AscDesc>Asc</AscDesc>
    </OrderBy>
    <OrderBy>
        <Field>S1.Cnty_Id</Field>
        <AscDesc>Asc</AscDesc>
    </OrderBy>
    <OrderBy>
        <Field>S1.City_Sect_Id</Field>
        <AscDesc>Asc</AscDesc>
    </OrderBy>
    <OrderBy>
        <Field>S2.StreetName</Field>
        <AscDesc>Asc</AscDesc>
    </OrderBy>
    <OrderBy>
        <Field>S2.Cnty_Id</Field>
        <AscDesc>Asc</AscDesc>
    </OrderBy>
    <OrderBy>
        <Field>S2.City_Sect_Id</Field>
        <AscDesc>Asc</AscDesc>
    </OrderBy>

```

```

        </OrderBy>
    </OrderByList>
    <UnionQueries>
    </UnionQueries>
    </QueryDefinition>
</UnionQueries>
</QueryDefinition>
<CalculatedFields/>
</pdroot>

```

HCL query

The HCL query is used by the High Crash Location report to query all of the records with in a date range. This query is used to retrieve all records between the first crash date entered and the last date entered by the user. It requires the parameters of :FirstDate AND :LastDate. The HCL query must contain all of the fields listed in your Default query to operate correctly.

This query is normally a copy of your Default query less any references to the primary street and cross street in the where clause. If your Default query look like this:

```

Select (list of fields)
FROM
    ( Dbo.Crashes Veh_1
      LEFT JOIN Dbo.Crashes Veh_2 ON ((Veh_2.Accident_Number=Veh_1.Accident_Number) AND (Veh_2.Veh
      LEFT JOIN Dbo.Crashes Veh_3 ON ((Veh_3.Accident_Number=Veh_1.Accident_Number) AND (Veh_3.Veh
WHERE
    (Veh_1.Vehicle_Number = 1)
    AND((( Veh_1.Primary_Road = :PrimaryStreet) AND
    ( Veh_1.Secondary_Road = :CrossStreet )) OR
    (( Veh_1.Primary_Road = :PrimaryStreet2 ) AND
    ( Veh_1.Secondary_Road = :CrossStreet2 )))
    AND((Veh_1.Collision_Date BETWEEN :FirstDate AND :LastDate))

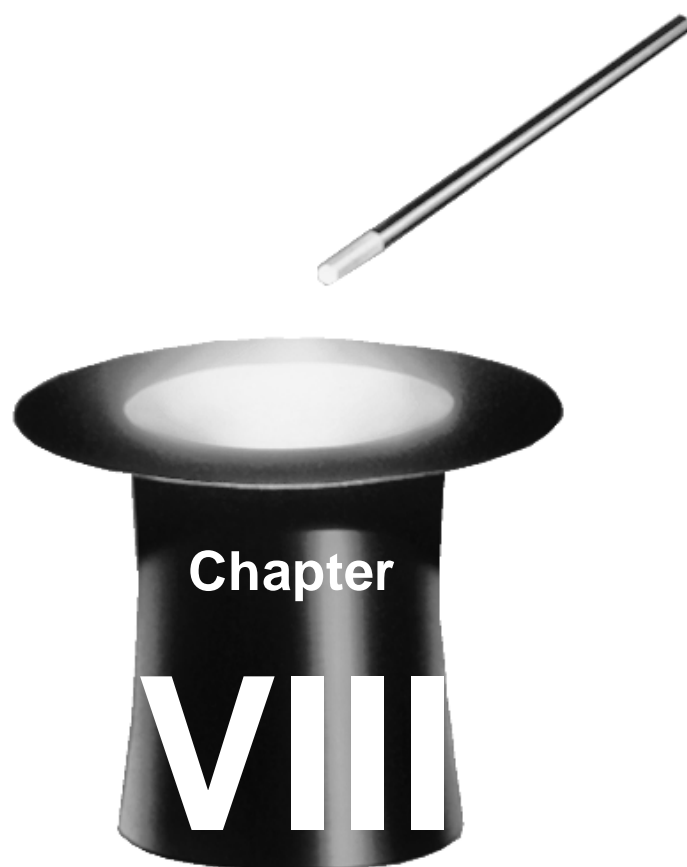
```

The Default query can be copied and renamed HCL. Then remove the AND section of the where clause that contains the

```

Select (list of fields)
FROM
    ( Dbo.Crashes Veh_1
      LEFT JOIN Dbo.Crashes Veh_2 ON ((Veh_2.Accident_Number=Veh_1.Accident_Number) AND (Veh_2.Veh
      LEFT JOIN Dbo.Crashes Veh_3 ON ((Veh_3.Accident_Number=Veh_1.Accident_Number) AND (Veh_3.Veh
WHERE
    (Veh_1.Vehicle_Number = 1)
    AND((Veh_1.Collision_Date BETWEEN :FirstDate AND :LastDate))

```



8 Automation and interoperability

8.1 Overview

In addition to be used interactively, Crash Magic can be used to generate reports as requested by other applications. The following are the supported methods for accessing and controlling Crash Magic programmatically:

- **Basic URL** - A simple URL interface for accessing program information without logging in. (all calls are prefixed with "\$")
- **MagicAuto** - A more advanced URL interface, requiring login, that provides a high level access to navigation and report generation by manipulating URL parameters.
- **Remote Access (REST)** - This method provides much more extensive control of the program including error handling and more granular access to program components.

Note: When utilizing any of these URI - based options, the first step is always gaining access to the server in the first place. If the server is already exposed, this will not require any additional steps. However, if the server is behind a firewall or protected by an authentication mechanism, this will need to be addressed. The first test for access to the system should be to check the version. This must be successful before attempting to make calls that require log in or additional data exchange. There are two mechanisms for this. One is in Basic URL and the other is in Remote access:

This first call, using Basic URL, will return just the simple text of the version, such as 4.1.172.0:

[https://www.governmenttools.com/demo/cmo.dll/\\$Version](https://www.governmenttools.com/demo/cmo.dll/$Version)

This second call, using Remote access, will return a more complete response in JSON, XML or HTML. The call below returns the version in a JSON object like this:

```
{
  "Version": "4.1.172.0",
  "Function": "VersionNH",
  "Method": "GET",
  "Application": "Crash Magic Online",
  "Server": "CMHOSTEDSERVER",
  "ServerVersion": "4.1.172.0",
  "Instance": "demo",
  "APIVersion": "4.0",
  "RequestBegin": "2020\\06\\12 12:05:19 170",
  "RequestEnd": "2020\\06\\12 12:05:19 170",
  "RequestElapsedMS": 0
}
```

<https://www.governmenttools.com/demo/cmo.dll/REST/4.0/VersionNH?rf=json>

Active Directory and URI access, such as REST

If your Crash Magic system is being protected via IIS Authentication such as Active Directory authentication, making these calls require the additional step of logging in. This will require an account that IIS will authenticate. Frequently, Active Directory

accounts, being set up for "real" users, are configured to expire their passwords periodically. This can be very inconvenient to REST programmers attempting to communicate with Crash Magic. These accounts should be set to "password never expires" if they will be used here.

Another option is to merely use a local server account. A user account created on the same server that hosts IIS will also authenticate. In fact, the user need not be added to any AD groups, as it will not need to do anything other than authenticate its password. For this reason, the Crash Magic installer creates an appropriate "cmREST" account during installation. Simply specify a password and press the "Create local cmREST user" button.

Crash Magic Online Install and Configuration Tool v1.2.1.22696

File installer | Pre-load settings | IIS Configuration | GIS IIS Configuration

This page is used for configuring the IIS Server to host Crash Magic as an existing or new site and/or application. First select the "instance to configure". Be sure that IIS is installed with HTTP Redirect and both ISAPI and .NET features included. See program manual for details

New Instance Configuration Steps

1. Select instance to configure:

Account and Folder Security

Group:

User:

Password:

2. Create this user and group

3. Give this user required access

Password:

optional: Create local cmREST user

IIS Site Settings

4. Select web site: New ...

5. Select application: New ...

6. Configure site and app:

7. Select application pool: New ...

8. Configure app pool:

Status [Clear](#)

Ready

Deprecated

A deprecated function will be removed in future versions of the program / API and should not be used in new code.

8.2 Basic URL

Crash Magic supports a number of requests that can be made using a URL. These calls do not require a login and are generally used for debugging and process monitoring.

\$DIAG

The \$DIAG function returns an information screen that can run a number of tests and set a couple of debug parameters.

```
/$DIAG?level=nn
```

[level=0 - Echo](#)

[level=1 - Info only](#)

[level=2 - Request](#)

[level=3 - Event log](#)

[level=4 - Exception](#)

[level=5 - Log file](#)

[level=6 - Web server settings](#)

[level=7 - CM server settings](#)

[level=8 - cmSysTables](#)

StatusLogLevel = [Detail](#) | [Debug](#) | [Info](#) | [Important](#) | [Warning](#) | [Error](#) | [Ex](#)

DebugSysSQL (False) = [true](#) | [false](#)

DebugUserSQL (False) = [true](#) | [false](#)

\$DIAG?Level=<n>

Level is parameter that may be specified to request a number of levels of operation of the running system

Possible values for "n" are:

1. Information (Process, memory load, ISAPI Pool parameters)
2. Request information (URL, path, etc.)
3. Write to Windows Event Log
4. Raise application exception (currently disabled)
5. Write to log file (debug, important, error)
6. Web server settings
7. Server settings
8. System tables - Attempts to use system tables manager to access .Master user group.

Example:

[http://www.governmenttools.com/demo/cmo.dll/\\$DIAG?Level=5](http://www.governmenttools.com/demo/cmo.dll/$DIAG?Level=5)

\$DIAG?

Set a new value for StatusLogLevel. Possible values for <level>

StatusLogLevel=<level> are:

> Detail, Debug, Info, Important, Warning, Error, Exception, None

\$DIAG? Turn on or off System SQL debugging

DebugSysSQL=true/false

se

\$DIAG? Turn on or off User SQL debugging

DebugUserSQL=true/false

alse

\$DIAG? Set the maximum number of ISAPI threads between 3 and 100.

MaxNumOfThreads=n

\$DIAG? Turns on Autostart. Useful when system started while missing resources such as dB connections

ResetAutoStart

\$Version

Returns the version of the running instance of Crash Magic.

[http://www.governmenttools.com/demo/cmo.dll/\\$Version](http://www.governmenttools.com/demo/cmo.dll/$Version)

4.1.174.0

\$SYSCONNECTION	Reports whether a connection to the system database is available http://www.governmenttools.com/demo/cmo.dll/\$SysConnection true false
\$SYSTABLE	Reports whether a query can be executed against the system's user group table http://www.governmenttools.com/demo/cmo.dll/\$SysTable true false

In addition to the standalone URL parameters, the MagicAuto functionality provides for system automation through the URL.

More advanced functionality is available using the "Remote access" interface

8.3 MagicAuto

MagicAuto is a URL-based command line interface to Crash Magic. MagicAuto requires that a user session be active before making any calls. That user session defines available projects and is used to regulate security access.

Note: It is also possible to pass login/password information, which can be encrypted, but the content is sent in clear text and is not recommended. See [Automatic login](#)³⁵⁰.

All MagicAuto GET calls are formatted as follows:

```
<ApplicationURL>/MagicAuto/Action=<Action>[?<Param>=<Value>]
[&<Param>=<Value>]...
```

For example:

http://demo.governmenttools.com/CrashMagicOnline_ISAPI.dll/MagicAuto?Action=home

or

http://demo.governmenttools.com/CrashMagicOnline_ISAPI.dll/MagicAuto?Action=goto&project=OldMain

Using GET or POST

When using MagicAuto, GET and POST are processed in the same way. This means that you can pass data and parameters to Crash Magic using URL parameters, or by using a form that posts to Crash Magic. The only major difference is that most browsers limit the amount of content that can be sent in the URL. (somewhere around 2000 characters for most browsers) POST does not have these limitations.

Actions

Parameters supported for an external call to Crash Magic:

Parameter	Description:	Supported values:
Action (required)	The Action parameter tells Crash Magic what to do. It determines what other parameters will be required and supported.	<ul style="list-style-type: none"> • home • get • set • goto <p><i>The following report types can be used as actions as well:</i></p> <ul style="list-style-type: none"> • diagram • chart • list • locationlist • layout • crosstab

Automatic login

Automatic login parameters (optional):

If the existing session login matches these login parameters, the session is not affected. If the session login name or group differ, the existing session is terminated and the login page is presented.

If no session exists, the login parameters are used to create a new session and automatically log in to the system.

Parameter	Description:	Supported values:
UserGroup	The user group to log into	Any existing user group in the system
UserLogin	The user to log in as	Any existing user in the specified user group
UserPassword	The user password	<p>The password that corresponds to the specified user and user group</p> <p>This password must be encrypted when encryption has been enabled. Crash Magic uses the current</p>

		<p>Greenwich Mean Time to expire encrypted passwords. Once encryption has been enabled any passwords sent to Crash Magic must have "=(The current GMT)" appended to the password before being encrypted and sent to Crash Magic. The password will not work if the time sent is more than one day off from the GMT of the server running Crash Magic. The format for the date is CCYY-MM-DD HH:MM:SS with the time in a 24hr format. If the current GMT is "10/31/2007 9:57:46 PM" and the password you are sending is "TestPassword" you must</p> <p>encrypt:TestPassword=2007-10-31 21:57:46</p>
--	--	---

Home

home:

Home ignores any other parameters sent. (exception: when home is called, login information will be applied as documented) Calling the program with Action=HOME causes the current user's home page to be displayed. This parameter is most often used to test the MagicAuto feature of the program.

GoTo

Goto is used to automatically navigate to a specific location in the project. At this time, that location may be specified by project, study and report values. (including report type and current tab). In the future this feature will be enhanced to support access into the filter editor, the field list editor, and other pages not directly referenced in the project tree.

Accessing Project, Study and Report locations:

When Action=GoTo, the most common parameters to follow will specify a Project, or a Project + Study or a Project + Study + Report in order to navigate to a specific tree location.

Parameter	Description:	Supported values:
-----------	--------------	-------------------

Project	The Project parameter tells Crash Magic which project to load. Since this project will be loaded, it must exist already.	Any existing project name
Study	The Study parameter tells Crash Magic which study to load. Since this study will be loaded, it must exist already.	Any existing study name
Report	The Report parameter tells Crash Magic which report to load. Since this report will be loaded, it must exist already.	Any existing report name
ReportType (optional)	The type of report to load. If there is only one report with the specified report name (as indicated by the "Report" parameter) then this parameter is optional. If more than one report exists with this name, such as a "Default" diagram and a "Default" chart, this parameter is required in order to resolve to the correct report.	<ul style="list-style-type: none"> • diagram • chart • list • gis • locationlist • layout

Example:

http://demo.governmenttools.com/CrashMagicOnline_ISAPI.dll/MAGICAUTO?Action=GoTo&project=OldMain&Study=SANTA+MONICA+BL+%26+WESTWOOD+BL&Report=Four+way+intersection&ReportType=diagram

Note that the ReportType is required, since multiple reports in a study may share the same name

Specifying just a project, or just a project and study is a valid method of opening a project or study without opening a report.

Accessing other locations:

Parameter	Description:	Supported values:
Page	The Page parameter provides access to specific, pre-defined pages in the system.	<ul style="list-style-type: none"> • Administration - (if security permits, loads the admin page) • Help - Raises the help dialog

		<ul style="list-style-type: none"> • Home - Loads current project home page
--	--	--

Example:

http://demo.governmenttools.com/CrashMagicOnline_ISAPI.dll/MagicAuto?Action=GoTo&page=help

Report creation

Report Type Actions

By specifying a report type as the action, a report of that type will be created. The valid actions / report types are:

- diagram
- chart
- list
- crosstab
- locationlist

Creating a report requires a study for the report to be attached to. When creating a report using MagicAuto, all the properties needed to create this study are required. The first property needed is the study definition.

Required study-related parameters:

Parameter	Description:	Supported values:
StudyDefinition	The study definition for the study. Available definitions are listed in the study settings panel as well as the study definition editor on the administration side of the program.	Any valid, existing study definition name in the configuration.

Each study definition is of a specific study type. Those types are listed across the top of the table below. The available properties are listed down the left side and mark as required or not with a black square.

	Case Id	Case Id secondary	Date Range	Intersection	Intersection Corridor	Node	Route Milepost	Street Address	X Y Rectangle	User Case ID Reference	User Case ID Reference 2	Case ID List*	Case ID List secondary*
Caseld	■												
CaseldSecondary		■											
FirstDate			■	■	■	■	■	■	■				
LastDate			■	■	■	■	■	■	■				
PrimaryStreet				■	■		■	■					
CrossStreet				■									
FirstCrossStreet					■								
LastCrossStreet					■								
XMin									■				
XMax									■				
YMin									■				
YMax									■				
Node						■							
UserCID1										■	■		
UserCID2											■		
FirstAddress								■					
LastAddress								■					
FirstMilepost							■						
LastMilepost							■						
CorrMaxFtFromEndpoints					■								
CorrMaxFtFromPrimary					■								
CorrMinFtFromInts					■								

*Studies do not contain Case Id list values. The values are stored in the CID table.

Case Id List and Case Id List Secondary report types require special treatment as described in the section "[Sending a Case Id List](#)".

Optional study-related parameters:

Parameter	Description:	Supported values:
Filter	Set the Filter to be used in this new study. If Filter is not specified, a <i>clear</i> filter is used.	Any valid, existing filter name.

Categories	Specifies the name of the Category list to be used with this new study. If Categories is not specified, a <i>none</i> categories will be used.	Any valid Category list name
<Filter Categories> FromStudy	Any study parameters can be copied from an existing study.	Any valid Project and Study pair separated by a comma. For example: FilterFromStudy=Main,Broadway+Main+St

Optional report-related parameters:

Template	Template causes the specified template name to be loaded into the new report. The default for a template, if not specified, is the user's default template for the specified report type.	Any valid, existing report template name that matches the report type requested.
----------	---	--

Optional project tree-related parameters:

If any of these parameters are omitted, the program will choose appropriate defaults.

Parameter	Description:	Supported values:
Project	<p>The Project parameter tells Crash Magic what project to use when creating a new diagram.</p> <p>Any valid project name may be specified. If the specified project does not exist, it will be created.</p> <p>If Project is not specified, then the "MagicAuto" project will be used. MagicAuto will be created if it does not already exist in the current user's project tree.</p>	Any valid project name
Study	This will be the name of the study that will be created for this diagram. If Study is omitted, it	Any valid study name

	<p>will default to the name specified in the "Name" parameter.</p> <p>If the specified study already exists, and has the same query, parameters, filter, etc. a new study will not be created.</p> <p>If the study doesn't exist it will be created. If it exists with different parameters, the name will be made unique and a new study will be created.</p> <p>If no "Name" parameter is given, and no Study is specified, then a unique name will be created for the study.</p>	
Report	<p>This will be the name of the diagram report that will be created. If Report is omitted, it will default to the name specified in the "Name" parameter.</p> <p>The specified name will be checked to be sure it does not already exist. If it does exist, it will be made unique by adding a number to it. i.e. My Diagram (1)</p> <p>If no "Name" parameter is given, and no Report is specified, then a unique name will be created for the diagram.</p>	Any valid report name
Name	<p>Name can be used to specify the same name for both the Study and Report. By specifying Name, and not specifying either Study or Report, Name will be used for both parameters.</p>	Any valid Study and Report name.

Case Id List study types query the crash data by requesting a specific list of Case Id's. (rather than the bulk of the study types that query based on other crash data values such as street names, dates, milepost, etc.) This is done by maintaining a table in the Crash Magic system tables containing the case id numbers and associated study id. As a result, unlike all other study types, the parameters needed for the query are not located in the study record. For this reason, creating a Case Id list study is handled a bit differently. While it is possible to create such a study using a URL command, the recommended method is to post the data from an html file.

As with other report creation described in this section, values must be sent for Action and StudyDefinition. What is different is that a parameter called LoadCID must be passed as well. It must contain a very specific XML format. That format looks like this:

```
<pdroot>
  <CIDList>
    <CID>
      <UniqueNum1>1100002</UniqueNum1>
    </CID>
    <CID>
      <UniqueNum1>1100006</UniqueNum1>
    </CID>
    <CID>
      <UniqueNum1>1100009</UniqueNum1>
    </CID>
    <CID>
      <UniqueNum1>1100010</UniqueNum1>
    </CID>
  </CIDList>
</pdroot>
```

If the Case Id values are not numeric, then the data elements will be called <UniqueStr> rather than <UniqueNum1>. Something like this:

```
<pdroot>
  <CIDList>
    <CID>
      <UniqueStr>14-1100002</UniqueStr>
    </CID>
    <CID>
      <UniqueStr>14-1100006</UniqueStr>
    </CID>
    <CID>
      <UniqueStr>14-1100009</UniqueStr>
    </CID>
    <CID>
      <UniqueStr>13-1100010</UniqueStr>
    </CID>
  </CIDList>
</pdroot>
```

So, a typical html file might look like this. Actually, this is a real file sent from our Map Magic plugin to ArcMap to generate a diagram.

```

<!-- saved from url=(0014)about:internet -->
<html>
<title>Crash Magic Online</title>
<body onload="document.autosubmit.submit();">
<body>
<form name="autosubmit" method="post" target="_self"
action="http://www.governmenttools.com/cm/CrashMagicOnline_ISAPI.dll/MagicAuto/
?ACTION=diagram&StudyDefinition=AccidentNbr List&Name=Selection%20Report">

<table width="100%" bgcolor="silver"><tr><td align="center">
<h1>Map Magic</h1>
Is preparing your new report<br/><br/><br/>
<input name="LoadCID" type="hidden"
value="<pdroot><CIDList>
<CID><UniqueNum1>1100002</UniqueNum1></CID>
<CID><UniqueNum1>1100006</UniqueNum1></CID>
<CID><UniqueNum1>1100009</UniqueNum1></CID>
<CID><UniqueNum1>1100010</UniqueNum1></CID>
<CID><UniqueNum1>1100012</UniqueNum1></CID>
<CID><UniqueNum1>1100013</UniqueNum1></CID>
<CID><UniqueNum1>1100020</UniqueNum1></CID>
<CID><UniqueNum1>1100021</UniqueNum1></CID>
<CID><UniqueNum1>1100022</UniqueNum1></CID>
<CID><UniqueNum1>1100023</UniqueNum1></CID>
<CID><UniqueNum1>1216487</UniqueNum1></CID>
<CID><UniqueNum1>1216488</UniqueNum1></CID>
</CIDList></pdroot>"
>
</form>
</tr></table>
<script language="Javascript">window.focus();</script>
</body>
</html>

```

Important: This section describes deprecated functionality that will be removed in future versions of Crash Magic. It is provided as a reference for programmers porting their old code to this new API.

Prior versions of Crash Magic populated the SQL query directly. This required knowledge of the data type, parameter and an index to tie them all together. This method of specifying query parameters is still supported, but will be removed in future versions of Crash Magic.

Also, the names for the Filter, categories and other parameters included the suffix "template" which is no longer required, but still supported.

chart, diagram, gis, layout, list, locationlist:

Parameter	Description:	Supported values:
Studydefinition (required)	The Studydefinition specifies which study definition to use.	Any valid study definition name in the configuration.

SQL parameter list (required)	<p>List of sql parameter names, types and values required for the specified query.</p> <p>SQLn=<parameter name> <parameter name>=<parameter value> <parameter name>T=<parameter type></p> <p>For example: SQL0=PrimaryStreet PrimaryStreet=Main Street PrimaryStreetT=string SQL1=FirstDate FirstDate=01/01/2005 FirstDateT=DateTime SQL2=LastDate LastDate=01/01/2005 LastDateT=DateTime SQL3=Distance Distance=200 DistanceT=Integer</p> <p>SQL0=PrimaryStreet&PrimaryStreet=Main Street&PrimaryStreetT=string& SQL1=FirstDate&FirstDate=01/01/2005&FirstDateT=DateTime& SQL2=LastDate&LastDate=01/01/2005&LastDateT=DateTime& SQL3=Distance&Distance=200&DistanceT=Integer</p>	<p>Parameter name must be the name of a parameter that exists in the query specified by the Query parameter. All parameters in the Query must be specified in the SQL parameter list for the study to load correctly.</p> <p>The starting SQL parameter is always SQL0.</p> <p>Parameter value must be the value to be passed to the query.</p> <p>Parameter type must be one of the types listed below. The type must match, or at least be compatible with the field type in the SQL database. Not all types are valid for all databases.</p> <p><i>The most common types are bold in the list. It is unlikely that you will have cause to use a different type than the common ones.</i></p> <table><tr><td>String</td><td>Smallint</td></tr><tr><td>Integer</td><td>Word</td></tr><tr><td>Boolean</td><td>Float</td></tr><tr><td>Currency</td><td>BCD</td></tr><tr><td>Date</td><td>Time</td></tr><tr><td>DateTime</td><td>Bytes</td></tr><tr><td>VarBytes</td><td>AutoInc</td></tr><tr><td>Blob</td><td>Memo</td></tr><tr><td>Graphic</td><td>FmtMemo</td></tr><tr><td>ParadoxOle</td><td>DBaseOle</td></tr><tr><td>TypedBinary</td><td>Cursor</td></tr><tr><td>FixedChar</td><td>WideString</td></tr><tr><td>Largeint</td><td>ADT</td></tr><tr><td>Array</td><td>Reference</td></tr><tr><td>DataSet</td><td>OraBlob</td></tr></table>	String	Smallint	Integer	Word	Boolean	Float	Currency	BCD	Date	Time	DateTime	Bytes	VarBytes	AutoInc	Blob	Memo	Graphic	FmtMemo	ParadoxOle	DBaseOle	TypedBinary	Cursor	FixedChar	WideString	Largeint	ADT	Array	Reference	DataSet	OraBlob
String	Smallint																															
Integer	Word																															
Boolean	Float																															
Currency	BCD																															
Date	Time																															
DateTime	Bytes																															
VarBytes	AutoInc																															
Blob	Memo																															
Graphic	FmtMemo																															
ParadoxOle	DBaseOle																															
TypedBinary	Cursor																															
FixedChar	WideString																															
Largeint	ADT																															
Array	Reference																															
DataSet	OraBlob																															

		OraClob Interface Guid	Variant IDispatch TimeStamp
CategoriesTemplate (optional)	Specifies the name of the Category list to be used with a study.	Any valid Category list name	
Project (optional)	<p>The Project parameter tells Crash Magic what project to use when creating a new diagram.</p> <p>Any valid project name may be specified. If the specified project does not exist, it will be created.</p> <p>If Project is not specified, then the "MagicAuto" project will be used. MagicAuto will be created if it does not already exist in the current user's project tree.</p>	Any valid project name	
Study (optional)	<p>This will be the name of the study that will be created for this diagram. If Study is omitted, it will default to the name specified in the "Name" parameter.</p> <p>If the specified study already exists, and has the same query, parameters, filter, etc. a new study will not be created.</p> <p>If the study doesn't exist it will be created. If it exists with different parameters, the name will be made unique and a new study will be created.</p> <p>If no "Name" parameter is given, and no Study is specified, then a unique name will be created for the study.</p>	Any valid study name	

Report(optional)	<p>This will be the name of the diagram report that will be created. If Report is omitted, it will default to the name specified in the "Name" parameter.</p> <p>The specified name will be checked to be sure it does not already exist. If it does exist, it will be made unique by adding a number to it. i.e. My Diagram (1)</p> <p>If no "Name" parameter is given, and no Report is specified, then a unique name will be created for the diagram.</p>	Any valid report name
Name (optional)	Name can be used to specify the same name for both the Study and Report. By specifying Name, and not specifying either Study or Report, Name will be used for both parameters.	Any valid Study and Report name.
Template (optional)	Template causes the specified template name to be loaded into the new report. The default for a template, if not specified, is the user's default template for the specified report type.	Any valid, existing template name.
FilterTemplate (optional)	FilterTemplate causes the specified filter template to be loaded into the study of the object created. The default is to load no filter.	Any valid, existing filter template name.
FilterFromStudy (optional)	FilterFromStudy causes the filter from the specified Project,Study to be copied to the new study being created.	Any valid Project and Study pair separated by a comma.

The following section breaks down some sample URL calls to Crash Magic. Specific examples can be tested on client instances by logging into Crash Magic and pasting the

url into the address bar of the browser to confirm the correct action occurred.

The following example is an HTML tag to call Crash Magic, and create a [diagram](#)^[99] under an [intersection study](#)^[136]. If the study does not already exist it will be created with the diagram. For this example to work with your data, crash records must have the ability to be located by a street and cross street location, and the user must be logged into Crash Magic. The Intersection study requires 2 string parameters to define the street names. The intersection study definition under stand that Broadway and Main is the same intersection as Main and Broadway. Extra spaces have been inserted into this example to allow this document to wrap - they should not exist in a true URL. (intended spaces will be escaped with "+" symbols)

```
<a href="http://www.governmenttools.com/cm/CrashMagicOnline_ISAPI.dll/MAGICAUTO/?ProjectName=Grant St and West Pine Ave& ACTION=diagram& Project=Demo& studydefinition=Intersection& PrimaryStreet=Grant St& CrossStreet=West Pine Ave& FirstDate=01/01/2007& LastDate=12/31/2007& filter=clear& template=Four way intersection& Categories=None" target="" title="Create a Diagram">Diagram</a>
```

(spaces were added after all "&" in above url to enable wrapping and increase readability. Do not include spaces in actual url.)

Item	Description
http://www.governmenttools.com/cm/CrashMagicOnline_ISAPI.dll/MAGICAUTO	This is the URL of the Crash Magic server. This location will be specific to your installation of Crash Magic.
MAGICAUTO	This value tells Crash Magic that you will be making an automated call.
Name=Grant St and West Pine Ave	Name that will be given to any item that is created. This name may be modified if you are creating an item that already exists.
Action=DIAGRAM	The Action ^[350] is what Crash Magic is being asked to perform.
Project=Demo	Name of the project that will hold the study. If this project name does not already exist, then it will be created.
studydefinition=Intersection	Name of the study definition ^[280] to be used.
PrimaryStreet=Grant St	The PrimaryStreet property for the study
CrossStreet=West Pine Ave	The CrossStreet property for the study
FirstDate=01/01/2007	The FirstDate property for the study
LastDate=12/31/2007	The LastDate property for the study
filter=clear	The filter to be used with this study.
template=Four way intersection	The report template to be used with the report. In this case it is the diagram template for a four way intersection.
Categories=None	The categories to be applied to the study.

The following example is an HTML tag to call Crash Magic, and create a node study diagram. For this example to work with your data, crash records must have the ability to

be located by a specific node number that identifies their location. The user must be logged into Crash Magic for this call to create the diagram.

```
<a href="http://www.governmenttools.com/cm/CrashMagicOnline_ISAPI.dll/MagicAuto/?
ACTION=diagram& studydefinition=Node& Project=Demo& Node=9270&
FirstDate=01/01/2004& LastDate=12/31/2005& Name=S SANTA FE DR/W ALAMEDA AVE&
filter=bicycles& template=Four way intersection& Categories=Time of day"
Target = "CMO">Node study</a>
```

(spaces were added after all "&" in above url to enable wrapping and increase readability. Do not include spaces in actual url.)

Item	Description
http://www.governmenttools.com/cm/CrashMagicOnline_ISAPI.dll/MagicAuto/	This is the URL of the Crash Magic server. This location will be specific to your installation of Crash Magic.
MAGICAUTO	The Action ³⁵⁰ is that Crash Magic is being asked to perform.
Action=DIAGRAM	Name that will be given to any item that is created. This name may be modified by Crash Magic if an item that already exists in the project.
Project=Demo	The project that the diagram will be created under. If this project does not exist it will be created.
studydefinition=Node	The study definition ²⁸⁰ that will be used to gather the collision records.
Node=9270	The node value.
FirstDate=01/01/2004	The FirstDate value.
LastDate=12/31/2005	The LastDate value.
Name=S SANTA FE DR/W ALAMEDA AVE	The name that will be given to the study.
filter=bicycles	The filter template that will be applied to the study.
template=Four way intersection	The diagram template that will be used to create the diagram report.
CategoriesTemplate=Time of day	The category template that will be applied to the study.

8.4 Remote Access

Crash Magic provides a Remote Access interface for application programmers to automate and manipulate objects inside of Crash Magic. By exposing a well defined API from Crash Magic, application programmers and web developers can access functionality built in to Crash Magic. A rich suite of functionality is available. The Remote Access functions may simply get a piece of text, cause a data change in Crash Magic, or an entire PDF document may be created and placed in a specified folder.

As with interactive use of the program, general method of accessing the Remote Access functions is to:

1. Log in

2. Perform analysis or administrative functions
3. Log out

During the login call, a unique identifier is created for the Remote Access "session". This handle is valid until the log out call. Every Remote Access function requires that this handle be passed as the first parameter.

The Remote Access calls are divided into several interfaces. The general design is an admin interface, an analysis interface, and a login interface. This reflects the interactive operation of the program.

Data structures

The Crash Magic remote access interface utilizes several standard data structures to make transfer of data more manageable. These structures are all XML based, making them easy to create, consume and pass across remote access boundaries.

SQLObjectList format

Many Remote Access functions return a list of SQLObjects in XML format. That format will include the Name, ObjType and, if the object is a User, the Login value. The output will look similar to the following:

The list structure will be customized for the object type. So, for example, a list of Reports would look like this:

```
<ReportList>
  <Report>
    <Name>Report One</Name>
    <ObjType>diagram</ObjType>
  </Report>
  <Report>
    <Name>Time of day</Name>
    <ObjType>chart</ObjType>
  </Report>
</ReportList>
```

A list of studies might look like this:

```
<StudyList>
  <Study>
    <Name>HWY 23</Name>
    <ObjType>routemp</ObjType>
  </Study>
  <Study>
    <Name>HWY 23 at Broadway</Name>
    <ObjType>intersection</ObjType>
  </Study>
</StudyList>
```


IcmRA_Admin

IcmRA_Admin provides access to all functions involving the creation, modification and deletion of User Groups, Users, and PSRattr templates.

Note: All of these functions require that the logged in Remote Access user have administrative privileges for the user group.

Group management

```
SOAP
GroupAdd(      aHandle, aUserGroupName, aParameters, &aErrorMsg );
```

```
REST
/GroupAdd?Handle=<value>&UserGroup=<value>&Parameters=<value>
returns ErrorMsg
```

Description: GroupAdd() creates a new user group in the CrashMagic system. This function will fail if the group already exists.

aHandle The value created in the call to [Login](#)³⁸⁸() .

aUserGroupName The user group into which the new user will be created. This group must exist prior to the call.

aParameters Reserved for future use, must be blank

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: GroupAdd(IHandle, "DOT@NY","", IErrorMsg);

```
SOAP
GroupDelete( aHandle,
             aUserGroupName,
             &aErrorMsg );
```

```
REST
/GroupDelete&Handle=<value>&UserGroup=<value>
returns ErrorMsg
```

Description: GroupDelete() removes a user group from the CrashMagic system. Removing a user group removes all users along with their templates, projects, studies and reports for this group as well. This function will fail if the account does not exist on the system.

aHandle The value created in the call to [Login](#)³⁸⁸() .

aUserGroupName The user group that the user exists in.

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: GroupDelete(IHandle, "DOT@NY", IErrorMsg);

Not documented

Not documented

User management

SOAP

```
UserAdd(      aHandle,
              aUserGroupName, aUserFullName,
              aUserLogin, aUserPassword,
              aCanAdminGroup, aCanEditData,
              aCanLoginAsSOAP, aCanBeCloned,
              aParameters,
              &aErrorMsg );
```

REST

```
/UserAdd?Handle=<value>&UserGroup=<value>&UserFullName=<value>
      &UserLogin=<value>&UserPassword=<value>
      &CanAdminGroup=<value>&CanEditData=<value>
      &CanLoginAsSOAP=<value>&aCanBeCloned=<value>
      &Parameters=<value>
returns ErrorMsg
```

Description: UserAdd() creates a new account in the CrashMagic system. This function will fail if the login already exists in the specified group.

aHandle The value created in the call to [Login](#)³⁸⁸() .

aUserGroupName The user group into which the new user will be created. This group must exist prior to the call.

aUserFullName Is the name of the user for this account. This is simply a descriptive field not used in the authentication process.

aUserLogin The desired login for this new account.

aUserPassword The desired password for this new account.

aCanAdminGroup Determines if this account should have group administrative privileges .

aCanEditData Determines if this account should be allowed to access the Data Entry screens .

aCanLoginAsRA Determines if this account should allow Remote Access logins.

aCanBeCloned Determines if this account can be cloned (i.e. for use with [LoginAsClone](#)³⁸⁹)

aParameters Reserved for future use, must be blank

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: UserAdd(IHandle, "DOT@NY", "Richard Scott Miller", "Scott", "djdekksla=", true, false, false, false, IErrorMsg);

```
SOAP
UserClone( aHandle, aUserGroupName, &aUserLogin, &aErrorMsg );
```

```
REST
/UserClone?Handle=<value>&UserGroup=<value>&UserLogin=<value>
returns ErrorMsg
```

Description: UserClone() Creates a new user account with a unique login. The GroupName/Login must match an existing account in the system. That account will be cloned, including all projects, studies, reports and templates. Only the login name will be modified.

aHandle The value created in the call to [Login](#)³⁸⁸().

aUserGroupName The user group that the user exists in.

&aUserLogin The account to clone. This value will be modified to new (cloned) login.

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: UserClone(IHandle, "DOT@NY", LoginTemplate, IErrorMsg);

```
SOAP
UserDelete( aHandle,
            aUserGroupName, aUserLogin,
            &aErrorMsg );
```

```
REST
/UserDelete?Handle=<value>&UserGroup=<value>&UserLogin=<value>
returns ErrorMsg
```

Description: UserDelete() removes a user account from the CrashMagic system. Removing a user account removes all templates, projects, studies and reports for this account as well. This function will fail if the account does not exist in the specified group.

aHandle The value created in the call to [Login](#)³⁸⁸().

aUserGroupName The user group that the user exists in.

aUserLogin The login to be deleted.

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: UserDelete(IHandle, "DOT@NY", "Scott", IErrorMsg);

```
SOAP
UserModify( aHandle,
            aUserGroupName, aUserLogin,
```

```

        &aNewUserLogin, aFullUserName, aUserPassword,
        aCanAdminGroup, aCanEditData,
        aCanLoginAsSOAP, aCanBeCloned,
        &aErrorMsg );

REST
/UserModify?Handle=<value>&UserGroup=<value>&UserLogin=<value>
        &NewUserLogin=<value>&FullUserName=<value>&UserPassword=<value>
        &CanAdminGroup=<value>&CanEditData=<value>
        &CanLoginAsSOAP=<value>&CanBeCloned=<value>
returns NewUserLogin, aErrorMsg

```

Description: UserModify() is used to change the login, full name, password and admin status for the specified account. This function will fail if the account does not exist in the specified group.

aHandle The value created in the call to [Login](#)^[388]().

aUserGroupName The user group that the user exists in.

aUserLogin The existing login for this account.

&aNewUserLogin The desired login for this account. This value may return as a different name if the specified login contained invalid characters or if the specified login already exists.

aUserFullName Is the name of the user for this account. This is simply a descriptive field not used in the authentication process. If this parameter is left blank, no changes are made to the field.

aUserPassword The desired new password for this account. If this parameter is left blank, no changes are made to the field.

aCanAdminGroup Determines if this account should have group administrative privileges .

aCanEditData Determines if this account should be allowed to access the Data Entry screens .

aCanLoginAsRA Determines if this account should allow Remote Access logins.

aCanBeCloned Determines if this account can be cloned (i.e. for use with [LoginAsClone](#)^[389])

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: UserModify(IHandle, "DOT@NY", "Scott", "Richard", "Richard Scott Miller", "djdekslla=", false, false, false, false, IErrorMsg);

Not documented

PSRattr management

```

SOAP
PSRattrAdd(  aHandle, aUserGroupName, aUserLogin,
              aProjectName, aStudyName, aReportName, aReportType,
              aPSRattrName, aObjType, aParameters,

```

```

        &aErrorMsg );

REST
/PSRattrAdd?Handle=<value>&UserGroup=<value>&UserLogin=<value>
    &Project=<value>&Study=<value>&Report=<value>&ReportType=<value>
    &PSRattrName=<value>&ObjType=<value>&Parameters=<value>
returns ErrorMsg
returns ErrorMsg

```

Description: PSRattrAdd() creates a new PSRattr object to the CrashMagic system.

aHandle The value created in the call to [Login](#)³⁸⁸() .

aUserGroupName The user group into which the new PSRattr will be created. This group must exist prior to the call.

aUserLogin The user into which the new PSRattr will be created. This user must exist prior to the call.

aProjectName, aStudyName, aReportName, aReportType (optional) Most PSRattr objects reside with a user. However, they may also reside at any of these levels if specified.

aPSRattrName Name of the new PSRattr

aObjType Type of the new PSRattr

aParameters Reserved for future use, must be blank

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: PSRattrAdd(IHandle, "DOT@NY", "Scott", "", "", "", "",
"MyPSRattr", "diagram.template", "", IErrorMsg);

```

SOAP
PSRattrDelete( aHandle,
    aUserGroupName, aUserLogin,
    aProject, aStudy, aReport, aReportType,
    aPSRattr, aObjType,
    &aErrorMsg );

REST
/PSRattrDelete?Handle=<value>&UserGroup=<value>&UserLogin=<value>&Project=<value>&Study=<value>
    &Report=<value>&ReportType=<value>&PSRattr=<value>&ObjType=<value>
returns ErrorMsg

```

Description: PSRattrDelete() removes a PSRattr record CrashMagic system. This function will fail if the PSRattr does not exist in the specified group/user.

aHandle The value created in the call to [Login](#)³⁸⁸() .

aUserGroupName The user group that the PSRattr exists in.

aUserLogin The user group that the PSRattr exists in.

aProject, aStudy, aReport, aReportType (optional) Most PSRattr objects reside with a user. However, they may also reside at any of these levels if specified.

aPSRattr The PSRattr to be deleted.

aObjType The ObjType of the PSRattr to be deleted. Multiple PSRattrs may have the same name if they are of different type

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: PSRattrDelete(IHandle, "DOT@NY", "Scott", "", "", "", "",
"Default", "Diagram.template", IErrorMsg);

Not documented

Utility functions

```
SOAP
CurrentLogins( aHandle, &aLoginListXML, &aErrorMsg );
```

```
REST
/CurrentLogins&Handle=<Value>
returns LoginListXML, ErrorMsg );
```

Description: CurrentLogins() Lists all currently logged in accounts in the system. The logins are provided in an XML format.

aHandle The value created in the call to [Login](#)³⁸⁸().

&aLoginList The resulting list of logins. The output XML will look like this:

```
<CurrentLogins>
  <Login SessionId="dkeis8d7d7s">
    <Name>Pete</Name>
    <UserGroupId>4</UserGroupId>
    <UserId>1</UserId>
    <MachineName>Wizard</MachineName>
  </Login>
  <Login SessionId="dkedd899d7x">
    <Name>Scott</Name>
    <UserGroupId>4</UserGroupId>
    <UserId>2</UserId>
    <MachineName>Wizard</MachineName>
  </Login>
</CurrentLogins>
```

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: CurrentLogins(IHandle, ILogins, IErrorMsg);

```
SOAP
ExpirePSRattrCacheItem( aHandle, aUserGroupName, aLoginName,
                        aProjectName, aStudyName, aReportName, aReportType,
                        aName, aObjType,
                        &aErrorMsg );
```

```

REST
/ExpirePSRattrCacheItem?Handle=<value>&UserGroup=<value>&UserLogin=<value>
&Project=<value>&Study=<value>&Report=<value>&ReportType=<value>
&Name=<value>&ObjType=<value>
returns ErrorMsg

```

Description: ExpirePSRattrCacheItem() Flags a cached PSRattr item as expired and no longer available. This will cause the next request to load from the database instead of the cache.

aHandle The value created in the call to [Login](#)³⁸⁸() .

To indicate which PSRattr to expire, specify the following values. PSRattr's may be referenced to the shared user, a specific user, a project, study or report.

aUserGroupName, aUserLogin - These are required values

aProjectName, aStudyName, aReportName, aReportType - Any or all of these may be specified or left blank

aName, aObjType - These are required.

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: ExpirePSRattrCacheItem(IHandle, "DOT@NY", ".shared", "", "", "", "",
"MySchematic", "schematic", &aHandle)

```

SOAP
ListGroups( aHandle, &aGroupList, &aErrorMsg );

```

```

REST
/ListGroups?Handle=<value>
returns GroupList, ErrorMsg

```

Description: ListGroups() Lists all the user groups in the system. The group names are provided in a comma-delimited list.

aHandle The value created in the call to [Login](#)³⁸⁸() .

&aGroupList The resulting list of group names in a comma-delimited format.

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: ListGroups(IHandle, IListOfGroups, IErrorMsg);

```

SOAP
ListUsers( aHandle, aUserGroupName, &aLoginList, &aErrorMsg );

```

```

REST
/ListUsers?Handle=<value>&UserGroup=<value>
return LoginList, ErrorMsg

```

Description: ListUsers() Lists all the user logins in the specified user group. The logins are provided in a comma-delimited list.

aHandle The value created in the call to [Login](#)³⁸⁸() .

aUserGroupName The user group to query.

&aLoginList The resulting list of user logins in a comma-delimited format.

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: ListUsers(IHandle, "DOT@NY", IListOfUsers, IErrorMsg);

SOAP

```
ListPSRattrs( aHandle,
               aUserGroupName, aUserLogin,
               aObjType, aIncludeShared,
               &aPSRattrList,
               &aErrorMsg );
```

REST

```
/ListPSRattrs?Handle=<value>&UserGroup=<value>&UserLogin=<value>
&ObjType=<value>&IncludeShared=<value>
returns PSRattrListXML, ErrorMsg
```

Description: ListGroups() Lists all the user groups in the system. The group names are provided in a comma-delimited list.

aHandle The value created in the call to [Login](#)³⁸⁸() .

aUserGroupName The user group to query.

aUserName The user to query.

aObjType An optional PSRattr object type to limit the listing to.

aIncludeShared If true, the list of user PSRattr objects will be combined with any PSRattr objects available to that user through group shared and master shared users.

&aPSRattrList The resulting list of PSRattr names and object types in a comma-delimited format.

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: ListPSRattrs(IHandle, "DOT@NY", "Pete", "layout.template", true, IListOfPSRattrs, IErrorMsg);

SOAP

```
PSRattrXMLModify( aHandle, aUserGroupName, aUserLogin,
                  aProjectName, aStudyName, aReportName,
                  aReportType, aPSRattrName, aObjType, aXML,
                  &aErrorMsg );
```

REST


```

/PSRattrXMLModify?Handle=<value>&UserGroup=<value>&UserLogin=<value>
    &ProjectName=<value>&StudyName=<value>&ReportName
    &ReportType=<value>&PSRattrName=<value>&ObjType=<value>&XML=<value>
returns aErrorMsg

```

Description: PSRattrXMLModify() Sets the XML for the specified PSRattr.

aHandle The value created in the call to [Login](#)³⁸⁸() .

To indicate which PSRattr to modify, specify the following values. PSRattr's may be referenced to the shared user, a specific user, a project, study or report.

aUserGroupName, aUserLogin - These are required values

aProjectName, aStudyName, aReportName, aReportType - Any or all of these may be specified or left blank

aPSRattrName, aObjType - These are required.

aXML The XML to populate the specified PSRattr with.

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: PSRattrXMLModify(IHandle, "DOT@NY", "Scott", "", "", "", "",
 "MyDiagram", "diagram.template", "<pdroot/>",
 IErrorMsg);

See also "LoadPSRattrXML" in lcmRA_Analysis

No documented

No-handle functions

```

SOAP
LicensedToNH( &aName, &aErrorMsg );

```

```

REST
/LicensedToNH
returns Name, ErrorMsg

```

Description: LicensedToNH() returns the name of the registered licensee of Crash Magic without requiring a "handle" or login.

&aName is populated with a string representing the licensee. The output will be similar to: "City of Anchorage, AK".

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: LicensedToNH(IName, IErrorMsg);

```
SOAP
ListGroupsWithNH( &aGroupListXML, &aErrorMsg );
```

```
REST
/ListGroupsNH
returns GroupListXML, ErrorMsg
```

Description: ListGroupsNH() Returns a list of all the user groups in the system as an XML Document. No login handle is required. The list includes the names of the groups, their logos and logo background colors.

&aGroupListXML The resulting list of group names in XML format. A typical file snippet might look like this:

```
<Groups>
  <Group Name="IA@DOT">
    <LoginLogo>IADOT.JPG</LoginLogo>
    <LoginLogoBgColor>#CC00CC</LoginLogoBgColor>
  </Group>
  <Group Name="CO@Denver">
    <LoginLogo>DenverCO.JPG</LoginLogo>
    <LoginLogoBgColor>#AA00CC</LoginLogoBgColor>
  </Group>
</Groups>
```

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: ListGroupsNH(IListOfGroups, IErrorMsg);

```
SOAP
ServerTimeNH( &aDateTime, &aErrorMsg );
```

```
REST
/ServerTimeNH
Returns ServerDateTime, ErrorMsg
```

Description: ServerTimeNH() returns the current date and time on the Crash Magic server without requiring a "handle" or login.

&aDateTime The Date and time on the server, represented as a string at the moment of the call. The format will appear as: "yyyy-mm-dd hh:nn:ss".

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: ServerTimeNH(IDateTime, IErrorMsg);

```
SOAP
VersionNH( &aVersion, &aErrorMsg );
```

```
REST
/VersionNH
Returns Version, ErrorMsg
```

Description: VersionNH() returns the current version of the Crash Magic server without requiring a "handle" or login.

&aVersion is populated with a string representing the current version of the program. The output will be similar to: "3.1.0.0".

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: VersionNH(IVersion, IErrorMsg);

```
SOAP
VersionNHExt( aProduct, &aVersion, &aErrorMsg );
```

```
REST
/VersionNHExt&aProduct=<Value>
returns Version, ErrorMsg
```

Description: VersionNHExt() returns the current version of the Crash Magic server, or any Crash Magic utility/add-on product, without requiring a "handle" or login.

aProduct is the name of the Crash Magic utility whose version number is desired.

Current possible values for this request are:

- "" (blank) returns the version of Crash Magic (Equivalent to VersionNH)
- "IW" Returns the version of the built-in html generator
- "cmDataEntry.exe" Returns the version of the installed Crash Magic data entry application available for download
- "svgView.exe" Returns the version of the SVG viewer application available for download
- "MapMagiccmo.dll" Returns the version of the Map Magic dll for ArcGIS available for download.
- Any other value will return an error

&aVersion is populated with a string representing the current version of the program. The output will be similar to: "3.1.0.0".

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: VersionNHExt("MagMagiccmo.dll", IVersion, IErrorMsg);

Deprecated.

This function was designed to return available Silverlight/XAML viewers for various data/report types.

IcmRA_Analysis

IcmRA_Analysis provides access to the creation, modification and deletion of projects, studies and reports. It also provides access to printing and other reporting functions.

Project management

```

SOAP
ProjectAdd( aHandle,
            aUserGroupName, aUserName,
            &aName, aDescription, aObjType,
            aTemplate, aParameters,
            aUseExisting,
            &aErrorMsg );

REST
/ProjectAdd?Handle=<value>&UserGroup=<value>&UserName=<value>
    &Project=<value>&Description=<value>&ObjType=<value>
    &Template=<value>&Parameters=<value>&UseExisting=<value>
returns Project, ErrorMsg

```

Description: Creates a new project in the CrashMagic system under the current account. If Name already exists, a unique name will be created and returned in the Name parameter.

aHandle The value created in the call to [Login](#)³⁸⁸().

aUserGroupName, aUserName Identify where the project is to be located

aName Is the name for the new project. This parameter may be changed by the call depending on the value of aUseExisting.

aDescription A text description for this project.

aObjType Type of project. (currently reserved, must be "project")

aTemplate The name of a template to base this new project on. If blank, the Default template will be used.

aParameters A list of parameters to modify the current settings. If blank, no settings are modified.

aUseExisting If UseExisting is true and a project with the same name already exists, a new project will not be created, and no error will be returned. If UseExisting is false and a project with the same name already exists, then a unique project name will be created and returned in the Name parameter.

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: ProjectAdd(IHandle, "CO@Denver", "Pete",

```

        "My first project", "A project for all seasons", "project", "Favorite",
        "", true, IErrorMsg );
Example: ProjectAdd( IHandle, "CO@Denver", "Pete",
                    IProjectName, "A project for all seasons", "project", "", "", false,
                    IErrorMsg );

```

```

SOAP
ProjectDelete( Handle, aUserGroupName, aUserName, aProjectName, &ErrorMsg );

```

```

REST
/ProjectDelete?Handle=<value>&UserGroup=<value>&UserLogin=<value>&Project=<value>
returns ErrorMsg

```

Description: Deletes the specified project from the current account.

aHandle The value created in the call to [Login](#)³⁸⁸().

aUserGroupName and aUserName - required

Name Is the name of the project to delete.

&ErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

```

Example: ProjectDelete( IHandle, "CO@FortCollins", "Sandy", "My first project",
                    IErrorMsg );

```

Not documented

Not documented

Not documented

```

SOAP
ListProjects( aHandle,
              aUserGroupName, aUserName
              &aProjectList
              &aErrorMsg );

```

```

REST
/ListProjects&Handle=<value>&UserGroup=<value>&UserLogin=<value>
returns ProjectList, ErrorMsg

```

Description: ListProjects() lists all the projects for the current UserGroup and User.

aHandle The value created in the call to [Login](#)³⁸⁸() .

aUserGroupName, aUserName Are required values.

&aProjectList The list of all projects for the specified UserGroup/User pair. This will be in standard [SQLObjectList XML format](#)³⁶⁴.

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: ListProjects(IHandle, "IA@DOT", "Michael", IProjectList, IErrorMsg);

Study management

```
SOAP
StudyAdd( aHandle,
          aUserGroupName, aUserName,
          aProjectName,
          &aName, aDescription, aObjType,
          aTemplate, aParameters, aFilterTemplate,
          aUseExisting,
          &aErrorMsg );
```

```
REST
/StudyAdd?Handle=<value>&UserGroup=<value>&UserLogin=<value>
&Project=<value>&Study=<value>&Description=<value>&ObjType=<value>
&Template=<value>&Parameters=<value>&FilterTemplate=<value>
&UseExisting=<value>
returns ErrorMsg
```

Description: Creates a new study in the CrashMagic system under the current account and within the current project. If Name already exists, a unique name will be created and returned in the Name parameter.

aHandle The value created in the call to [Login](#)³⁸⁸() .

aUserGroupName, aUserName - required

aProjectName The name of the project into which this study will be placed.

&aName Is the name for the new study. This parameter may be changed by the call depending on the value of UseExisting.

aDescription A text description for this study.

aObjType Type of study. (study | intersection | node | milepost | marris | universal)

aTemplate The name of a template to base this new study on. If blank, the Default template will be used.

aParameters An XML data structure containing a list of parameters to include with the study. The number and types of parameters is determined by the query used by the study. For an intersection query that requires a date range, primary and cross streets, and the reversed version of the primary and cross streets, the parameters XML string will look like this:

```
<Parameters>
```

```

        <Parameter>
            <Name>FirstDate</Name>
            <ParamType>Date</ParamType>
            <ParamValue>10/26/2006</ParamValue>
            <Enabled>True</Enabled>
        </Parameter>
        <Parameter>
            <Name>LastDate</Name>
            <ParamType>Date</ParamType>
            <ParamValue>10/26/2007</ParamValue>
            <Enabled>True</Enabled>
        </Parameter>
        <Parameter>
            <Name>PrimaryStreet</Name>
            <ParamType>String</ParamType>
            <ParamValue>BROADWAY</ParamValue>
            <Enabled>True</Enabled>
        </Parameter>
        <Parameter>
            <Name>CrossStreet2</Name>
            <ParamType>String</ParamType>
            <ParamValue>BROADWAY</ParamValue>
            <Enabled>True</Enabled>
        </Parameter>
        <Parameter>
            <Name>CrossStreet</Name>
            <ParamType>String</ParamType>
            <ParamValue>MAIN ST</ParamValue>
            <Enabled>True</Enabled>
        </Parameter>
        <Parameter>
            <Name>PrimaryStreet2</Name>
            <ParamType>String</ParamType>
            <ParamValue>MAIN ST</ParamValue>
            <Enabled>True</Enabled>
        </Parameter>
    </Parameters>

```

aFilterTemplate The name of a filter template to be loaded into the study after it has been created.

aUseExisting If aUseExisting is true and a study with the same name already exists, with the same parameters and settings, a new study will not be created, and no error will be returned. If aUseExisting is false and a study with the same name already exists, then a unique study name will be created and returned in the Name parameter.

&ErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: StudyAdd(IHandle, "NY@DOT", "Don", "Project1", "My first study", "A study of important things", "intersection", "Default", "", "", "", true, IErrorMsg);

Example: StudyAdd(IHandle, "NY@DOT", "Don", "Project1", IStudy, "A study of more importance", "intersection", "Favorite1", "<Parameters/>", "", "", false, IErrorMsg);

```

SOAP
StudyDelete( Handle,

```

```
aUserGroupName, aUserName,
aStudyName,
&ErrorMsg );
```

```
REST
StudyDelete?Handle=<value>&UserGroup=<value>&UserLogin=<value>&StudyName=<value>
returns ErrorMsg
```

Description: Deletes the specified study from the current account.

aHandle The value created in the call to [Login](#)³⁸⁸().

aUserGroupName, aUserName - required

aStudyName Is the name of the study to delete.

ErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: StudyDelete(IHandle, "IA@DOT", "Ravi", "My first study", IErrorMsg);

not documented

not documented

not documented

```
SOAP
ListStudies( aHandle,
aUserGroupName, aUserName, aProjectName
&aStudyList
&aErrorMsg );
```

```
REST
/ListStudies?Handle=<value>&UserGroup=<value>&UserLogin=<value>&Project
returns StudyList, ErrorMsg
```

Description: ListStudies() lists all the studies for the current UserGroup, User and Project.

aHandle The value created in the call to [Login](#)³⁸⁸().

aUserGroupName, aUserName, Project Are required values.

&aStudyList The list of all studies for the specified UserGroup/User/Project pair. This will be in standard [SQLObjectList XML format](#)³⁶⁴.

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: ListStudies(IHandle, "AK@Anchorage", "Steve", "Automatic", IStudyList, IErrorMsg);

Report management

SOAP

```
ReportAdd( aHandle,
           aUserGroupName, aUserName,
           aProjectName, aStudyName,
           &aName, aDescription, aObjType, aTemplate, aParameters,
           aUseExisting, &aErrorMsg );
```

REST

```
/ReportAdd?Handle=<value>&UserGroup=<value>&UserLogin=<value>
&Project=<value>&Study=<value>&Report=<value>&Description
&ObjType=<value>&Template=<value>&Parameters=<value>&UseExisting=<value>
returns Report, ErrorMsg
```

Description: Creates a new report in the CrashMagic system under the current account and within the current project and study. If Name already exists *for this type of report*, a unique name will be created and returned in the Name parameter.

aHandle The value created in the call to [Login](#)³⁸⁸().

aUserGroupName, aUserName - required

aProjectName Is the name of the project that this report will be placed into.

aStudyName Is the name of the study that this report will be placed into.

&aName Is the name for the new report. This parameter may be changed by the call depending on the value of UseExisting.

aDescription A text description for this report.

aObjType Type of report. (diagram | chart | list | hcllist | map)

aTemplate The name of a template to base this new report on. If blank, the Default template will be used.

aParameters A list of parameters to modify the current settings. If blank, no settings are modified.

aUseExisting If UseExisting is true and a report with the same name and ObjType already exists, with the same parameters and settings, a new study will not be created, and no error will be returned. If UseExisting is false and a report with the same name already exists, then a unique report name will be created and returned in the Name parameter.

ErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: ReportAdd(IHandle, "NY@DOT", "Rob", "Project1", "My first study", "A new report", "A report description", "diagram", "Default", "", true, IErrorMsg);

Example: StudyAdd(IHandle, "NY@DOT", "Rob", "Project1", "Study1", IReport, "Another report", "diagram", "Favorite1", "", false, IErrorMsg);

```

SOAP
ReportDelete( Handle, aUserGroupName, aUserName,
              aReportName,
              &aErrorMsg );

```

```

REST
/ReportDelete?Handle=<value>&UserGroup=<value>&UserLogin=<value>&Report=<value>
returns ErrorMsg

```

Description: Deletes the specified report from the specified account.

aHandle The value created in the call to [Login](#)³⁸⁸().

aUserGroupName, aUserName - required

aReportName Is the name of the report to delete.

&ErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: ReportDelete(IHandle, "AZ@Glendale", "Dave", "My first study", IErrorMsg);

not documented

not documented

Enter topic text here.

```

SOAP
ReportPrintPDF( aHandle, aUserGroupName, aUserName,
                aProjectName, aStudyName, aReportName, aObjType,
                aLayoutTemplate, aFileName, &aErrorMsg : string )

```

```

REST
/ReportPrintPDF?Handle=<value>&UserGroup=<value>&UserLogin=<value>
                &Project=<value>&Study=<value>&Report=<value>&ObjType=<value>
                &LayoutTemplate=<value>
returns FileName, ErrorMsg

```

Description: Produces a PDF file based on the specified layout template.

aHandle The value created in the call to [Login](#)³⁸⁸().

aUserGroupName, aUserName - required

aProjectName Can be used to specify the default project.

aStudyName Can be used to specify the default study.

aReportName Can be used to specify the default report.

aReportType If aReportName is specified, this parameter identifies the report type.

aLayoutTemplate A layout template to render to PDF. If blank, the default template for the report type will be used.

aFileName The full path and file name where the PDF file should be created. This is relative to the server.

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: ReportPrintPDF(IHandle, "AK@Anchorage", "Beth", "My Project", "My Study", "My Report", "diagram", "MySideBySide", "c:\temp\pdfs\output.pdf", IErrorMsg);

SOAP

```
ListReports( aHandle,
             aUserGroupName, aUserName, aProjectName, aStudyName,
             &aReportList
             &aErrorMsg );
```

REST

```
/ListReports?Handle=<value>&UserGroup=<value>&UserLogin=<value>&Project=<value>&Study=<value>
returns ReportList, ErrorMsg
```

Description: ListReports() lists all the reports for the specified UserGroup, User, Project and Study.

aHandle The value created in the call to [Login](#)³⁸⁸().

aUserGroupName, aUserName, Project, Study Are required values.

&aReportList The list of all reports for the specified UserGroup/User/Project/Study. This will be in standard [SQLObjectList XML format](#)³⁶⁴.

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: ListReports(IHandle, "AK@Anchorage", "Steve",
"Automatic", "Main St at Broadway", ReportList, IErrorMsg);

```
TemplateGroupAdd( aHandle, aUserGroupName, aUserName,
                  aProjectName, aStudyName,
                  aTemplateName,
                  aErrorMsg );
```

Description: TemplateGroupAdd() applies a template group to the specified study. A template group is a list of report templates that will reflect the data in the specified study.

aHandle The value created in the call to [Login](#)³⁸⁸().

aUserGroupName, aUserName - required

aProjectName Can be used to specify the default project.

aStudyName Can be used to specify the default study.

aTemplateName The name of the TemplateGroup
&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: TemplateGroupAdd(IHandle, "IA@DOT", "Pete",
 "My Project", "My Study", "Diagram plus listing",
 IErrorMsg);

```
ReportContent( aHandle, aUserGroupName, aUserName,
               aProjectName, aStudyName, aReportName, aObjType, aOptions,
               &aContent, &aErrorMsg )
```

Description: Renders the specified report in the format specified in aOptions.

aHandle The value created in the call to [Login](#)³⁸⁸().

aUserGroupName, aUserName - required

The following values specify the location of the report to render:

aProjectName, aStudyName, aReportName, aReportType

aOptions reserved

&aContent The resulting rendered report. Currently only XAML is supported.

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: ReportContent(IHandle, "AK@Anchorage", "Beth",
 "My Project", "My Study", "My Report", "diagram", "",
 IRenderedContent, IErrorMsg);

Utility functions

```
SetProjectStudyReport( aHandle,
                       aProjectName, aStudyName,
                       aReportName, aObjType,
                       &aErrorMsg );
```

Description: Sets the current Project, Study and Report. A report requires a study, which in turn requires a project. A blank parameter clears that attribute.

aHandle The value created in the call to [Login](#)³⁸⁸().

aProjectName The desired current project.

aStudyName The desired current study.

aReportName The desired current report.

aObjType Type of report (see ReportAdd for details)

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: SetProjectStudyReport(IHandle, "My first project", "My first study", "My
 second diagram", "diagram", IErrorMsg);

Example (sets a current study): SetProjectStudyReport(IHandle, "My first project", "My first study", "", "", IErrorMsg);

Example (sets a current project): SetProjectStudyReport(IHandle, "My first project", "", "", "", IErrorMsg);

```
LoadPSRattrXML( aHandle, aUserGroupName, aUserName,
                aProjectName, aStudyName, aReportName, aReportType,
                aPSRattrName, aObjType,
                &aXML, &aErrorMsg )
```

Description: Returns the content of the specified PSRattr XML

aHandle The value created in the call to [Login](#)³⁸⁸().

aUserGroupName, aUserName - required

The following values may be used to locate the PSRattr, which may be associated with a user, project, study or report:

aProjectName, aStudyName, aReportName, aReportType

aPSRattrName The name of the PSRattr to load.

aObjType The type of the PSRattr to load

&aXML The XML content of the specified PSRattr

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

```
Example: LoadPSRattrXML( IHandle, "CO@Lafayette", "Dennis",
                        "", "", "", "",
                        "Int_4", "schematic", IErrorMsg )
```

IcmRA_Login

IcmRA_Login provides login and logout functions for Crash Magic.

In order to begin a Remote Access session, the first call must be to login. If the intent is to perform analysis, the login account should be for a user of the system in the appropriate group. If the login is successful, the Handle parameter will be populated with a handle to that login. In order for that account to be used again, that handle must be logged out first.

The log in step determines the access that this Remote Access session will have. A Remote Access session will not be able to perform administrative operations such as adding or deleting users unless the login used has Group administrator permission.

Important: In order to log a user in as a Remote Access session, that user must also have "Can Login as RA" (Remote Access) permission. This is required of

administrative accounts that desire Remote Access as well. *Note: in order to perform administrative functions, the specified login must have administrative privileges in that group.*

RA Login General Information

In order to use the Remote Access functions in Crash Magic, a login must be performed. This login requires a password to be sent to the system. In addition, when creating or modifying a user account, the password may need to be sent.

Since some methods of using Remote Access utilize a "clear text" transmission, SOAP for example, passwords can be encrypted when sent to Crash Magic Remote Access Routines. Note that encryption is not required, and unencrypted is certainly the best way to get started while becoming familiar with the Remote Access routines.

Support for password encryption is set at the user group level. To enable a Crash Magic user group to require encryption the encryption method should be entered in the settings tab of the user group in the Encryption Method box, and an Encryption Key entered. The supported encryption method is 3Desv1 with a 16 character key.

Crash Magic uses the current Greenwich Mean Time to expire encrypted passwords. Once encryption has been enabled any passwords sent to Crash Magic must have "=(The current GMT)" appended to the password before being encrypted and sent to Crash Magic. The password will not work if the time sent is more than a minute off from the GMT of the server running Crash Magic. If the current GMT is "10/31/2007 9:57:46 PM" and the password you are sending is "TestPassword" you must
`encrypt:TestPassword=10/31/2007 9:57:46 PM`

The LoginAsClone function is the preferred method of performing analysis using Remote Access. The LoginAsClone function makes a complete copy of a specified account including analysis reports. By using the DeleteUserRecord parameter when logging out, this cloned account is removed from the system and any added or modified content is discarded with it. Using this method prevents any changes to the account that is cloned, leaving it available for future use regardless of changes made after logging in as clone.

A handy way of performing analysis, and even administrative tasks, is to login as a clone of an existing user. This method of logging in creates a copy of a specified account that the programmer can use while creating projects, studies, reports, etc. When logging out at the end of a "clone" session, the programmer may opt to delete this clone account from the database, thus removing all projects, studies, reports, templates or anything else created in that account's project tree. This method of using the Remote Access engine in Crash Magic is especially useful in a multi-threaded environment such as when using Crash Magic to generate reports for another multi-user application such as a web server or client server network application. In this case, a single account is created

just for the purpose of cloning. That account is provided with appropriate templates, reports, etc. which are then copied for use by the clone.

In order to clone a user, that account must be flagged as "Can be cloned".

If you are using remote access to manipulate shared resources or accounts other than your own, you will need to log in as an administrator.

- Logging in as an administrator of your own user group: In this situation, the administrator may manipulate:
 - User accounts in this user group
 - Resources of users in this user group
 - Resources in the .shared account for this user group
 - **Special:** If the logged in user is listed in the "config editor" section of the .configs account, the user will be able to manipulate resources in the .config account for this user group.
 - **Special:** If the logged in user is listed in the "config editor" section of an inherited group's .configs account, the user will be able to manipulate resources in the .config account for that inherited user group as well
- Logging in as the administrator of a shared user group: This should never occur. There should be no user logins for a shared user group. In order to manipulate a shared user group, a user should be created in a group that inherits from that shared user group. That user should be added to the config editor section of the shared user group's .config account.
- Logging in as a master administrator. This option is only available for clients with Crash Magic installed on their own servers. Clients that are hosted at Pd' Programming will not have such an account. When logged in as a master administrator, with the "group administrator" permission enabled, the logged in user will have access to all groups and user accounts and all of their resources. Be careful with this functionality. In general, this means of accessing the system should only be needed by an application that creates and manages multiple user groups on the system. When only managing one user group, always log in to that user group. Special note: It is possible to create a master user that does not have the group administrator permission enabled

CheckLogin

SOAP

```
CheckLogin( aHandle, aUserGroupName, aUserLogin, aLogoutIfFound,  
            &Found, &aErrorMsg );
```

REST

```
/CheckLogin?Handle=<Value>&UserGroup=<Value>&UserLogin=<Value>&LogoutIfFound=<Value>  
Returns: Found, ErrorMsg
```

Description: CheckLogin() Searches currently active logins for specified UserLogin. Found returns true if login exists. Optionally, LogoutIfFound will logout this user if it exists.

aHandle The value created in the call to [Login](#)³⁸⁸() .

aUserGroupName The name of the group that the specified Login is a member of.

aUserLogin The account to login as.

LogoutIfFound logs out this user if found to be logged in. (requires that the current user is a group admin)

&Found returns true if the login if found, false if not.

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

CheckLoginNH

SOAP

```
CheckLoginNH( aUserGroupName, aUserLogin, aPassword, aLogoutIfFound,
              &Found, &aErrorMsg );
```

REST

```
/CheckLoginNH?Handle=<Value>&UserGroup=<Value>&UserLogin=<Value>&Password=<Value>&LogoutIfFound=<Value>
Returns: Found, ErrorMsg
```

Description: Same as RACheckLogin, but does not require a handle. However, requires a password.

aPassword Password for the specified login, in the specified group. The password field may be encrypted based on the encryption defined for the specified user group.

Login

SOAP

```
Login( aUserGroupName, aUserLogin, aPassword,
       &aHandle, &aErrorMsg );
```

REST

```
/Login?UserGroup=<Value>&UserLogin=<Value>&Password=<Value>
Returns: Handle,ErrorMsg
```

Description: Login() Creates an entry in the login table indicating that this user is accessing the system. No other systems may access this login until Logout() is called.

aUserGroupName The name of the group that the specified Login is a member of.

aUserLogin The account to login as.

aPassword Password for the specified login, in the specified group. The password field may be encrypted based on the encryption defined for the specified user group.

&aHandle is populated when this function returns successfully. It contains an encrypted token that identifies this login. Keep this token for all future calls, and for logging out.

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

LoginAsClone

```
SOAP
LoginAsClone( aUserGroupName, &aUserLogin, aPassword,
              &aHandle, &aErrorMsg );
```

```
REST
/LoginAsClone?UserGroup=<Value>&UserLogin=<Value>&Password=<Value>
Returns: Handle, UserLogin, ErrorMsg
```

Description: LoginAsClone() Creates a new User account with a unique name, based on the aUserLogin parameter. The unique name is returned by modifying the aUserLogin parameter. This new account is the logged in the same manner as [Login](#)³⁸⁸ ().

aUserGroupName The name of the group that the specified Login is a member of.

&aUserLogin The account to base the clone on. The return value of this field is the new unique login that is created in the clone process.

aPassword Password for the login to be cloned. This will also be the login for the clone. The password field may be encrypted based on the encryption defined for the specified user group.

&aHandle is populated when this function returns successfully. It contains an encrypted token that identifies this login. Keep this token for all future calls, and for logging out.

&aErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: LoginAsClone("DOT@NY", lLogin, "ADFdfFDaFDd=", lHandle, lErrorMsg);

Logout

```
SOAP
Logout( aHandle,
        aDeleteUserRecord,
        &aErrorMsg );
```

```
REST
/Logout?Handle=<Value>&DeleteUserRecord=<Value>
Returns: ErrorMsg ;
```

Description: Logout removes the login entry and frees any resources associated with this user.

aHandle The value created in the call to [Login](#)³⁸⁸ (). It is considered undefined after a call to SOAPLogout().

aDeleteUserRecord may be set to true in order to delete the user account from the system after the logout is complete. This parameter is ignored unless the account was created by [LoginAsClone](#)³⁸⁹. (As indicated by a login that starts with "_@_").

&ErrorMsg is populated only if an error occurred that prevented the function from succeeding. Otherwise, this parameter is empty.

Example: Logout(IHandle, IErrorMsg);

SaveState

Not yet documented

SetSettingString

Not yet documented

GetSettingString

Not yet documented

Languages

Crash Magic remote access calls can be made from any program that supports SOAP or REST

REST

Crash Magic supports a REST (REpresentational State Transfer) interface to provide access to the full RemoteAccess API.

The REST endpoint is <server>/REST/3.9S (i.e.
www.GovernmentTools.com/REST/3.9S

Version 3.9S

This version of the REST interface is a port of the existing SOAP interface. As a result, it utilizes URL parameters rather than paths to make function calls.

All REST calls return an ErrorMsg value indicating the success of the call. A non-blank value indicates an error.

All REST calls return the name of the function called as well as the server name.

The RF parameter indicates the desired return format (HTML, XML, JSON)

Example request with JSON response

Request:

```
http://demo.governmenttools.com/crashMagicOnline_ISAPI.dll/REST/Login?UserGroup=AZ@Mesa
```

Response:

```
{
  "Handle": "",
  "Error": "EcmRA_Base: Login\n EcmRA_Base: InternalDecryptPassword\n EcmRA_Base: Encry",
  "Function": "Login",
  "Server": "Crash Magic Online"
}
```

Example request with XML response

Request:

```
http://demo.governmenttools.com/crashMagicOnline_ISAPI.dll/REST/Login?UserGroup=AZ@Mesa
```

Response:

```
<pdroot>
  <Response>
    <Handle/>
```

```
<Error>EcmRA_Base: Login EcmRA_Base: InternalDecryptPassword EcmRA_Base: Encr
<Function>Login</Function>
<Server>Crash Magic Online</Server>
</Response>
</pdroot>
```

SOAP

SOAP or "Service Oriented Architecture Protocol" is a mechanism for communication between applications across a network. SOAP uses XML packets to make requests from any application to Crash Magic and then receive a response back.

All Remote Access functionality provided for Crash Magic is available through the SOAP interface.

The available interfaces are accessible from any running Crash Magic instance, from the WSDL URL. (i.e.

http://www.governmenttools.com/cm/CrashMagicOnline_ISAPI.dll/wsdl)

The available interfaces are:

- lcmRA_Login Provides log in and log out functions.
- lcmRA_Admin Provides User Group, User and PSRattr access routines
- lcmRA_Analysis Provides Project, Study and Report access routines
- lcmRA_ODesktop Provides access to the desktop interface. It should never be used with the server product

When calling Crash Magic SOAP functions from C#, the following calling conventions are required:

- All "var" parameters, or "pass by reference" calls, indicated in this documentation by preceding the variable name with a "&" must be treated as "ref" parameters in C#. This means that if a parameter is documented as &ErrorMsg, it must be initialized and used as shown:

```
string IVersion = "";
string IErrorMsg = "";
VersionNH( ref IVersion, ref IErrorMsg );
```

WSDL is the language used by development applications, such as Visual Studio or Delphi, to determine the functions available in Crash Magic. Most development environments are capable of importing WSDL and generating wrapper functions in the desired language which will access the SOAP routines.

The url for Crash Magic is simply the same url used to access the normal application, except instead of the "MAGIC" keyword on the end of the URL, the word "WSDL" should be placed there instead. For example:

http://www.governmenttools.com/cm/CrashMagicOnline_ISAPI.dll/wsdl

or, when Crash Magic is run as a Windows service:

<http://LanMachine/wsdl>

COM

COM is not a supported interface for Remote Access and there is no plan for it in future releases. Please contact us if this is a requirement of your project(s).

Version log

3.9.122

This is the first version to support REST. All of the API functions are now available as REST functions. This initial REST interface is a simple port of the SOAP interface, so it does not yet support a standard CRUD API. It also does not use sub directories or folders for REST navigation, instead it uses URL parameters

3.0.0

Starting with Crash Magic Version 3, the RemoteAccess calls no longer reference the "current" login, project, study, report, etc. After logging in, a remote access call may affect, and must specify, the usergroup, user, project, study, report to act on. This change makes it possible to fully multi-thread the RemoteAccess interface and makes it possible to affect the account of a currently logged-in user while they work. (an administrative login is required for this capability)

Index

- _ -

_UniqueFields Unique Field 221

- A -

Action (administration) 350

Active Directory login

Administration 196

Add to favorites 87

Add user

Installation 180

Address 5

Address study type 140

Ad-hoc query 76

Administration 147

Aliases

Duplicate 61

Intersections 59

Nodes 61

Street names 50

Visible 50

All data study type 140

Analysis

Tutorial 9

Annotate 66

Automation

Administration 346

- B -

Buttons 66

- C -

Cache

Administration 302

Case id study type 141

CfgFileName

Installation 149

CfgName

Installation 149

Chart

Layout elements 99

Tutorial 24

Chart settings 98

Charts 96

How do I 47

ClickOn

Tutorial 20

cmRA_Analysis (administration) 376

cmUser

Installation 149

Collision diagrams 99

How do I 47

COM (administration) 392

Command (administration) 350

Configuration 307

Configuration (administration) 322

Connection string

Installation 180

Connections (configuration creation) 227

Contact information 5

Coordinates 142

Copy 66

Copy and paste

Tutorial 35

copy button 66

Corridor Diagram (administration) 336

Corridor schematic 102

Crash data

Export 75

Installation 180

Crash details

Tutorial 20

Crash listing 106

Editor 106

Fields 106

Layout elements 108

Crash listing settings 105

CrashMagic.ini

Installation 149

CrashMagicAP

Installation 149

Custom help (administration) 193

Custom query 76

- D -

Data

- Data
 - Export 75
- Database connection
 - Installation 180
- Database functions 81
- Database pass-through login
 - Administration 195
- Database specification (administration) 185
- Date functions 80
- DBVALUE 275
- DBVerify login %USER.LOGIN% 229
- Delete 66
- delete button 66
- Diagram
 - Layout elements 103
 - Schematics 102
 - Tutorial 15
- Diagram settings 100
 - Tutorial 23
- Diagrams
 - How do I 47
- Directories
 - Administration 302
- Documentation 6
- Duplicate intersections 61

- E -

- Editors
 - Filter (tutorial) 38
 - Intersection aliases 61
 - Street aliases 51
- EMail 5
- Example filters 86
- Export 75
 - Crash data 75
 - Study 75

- F -

- Favorite 87
- Fax 5
- Fields
 - Crash listing 106
- Files
 - Administration 302
 - CrashMagic.ini 149

- Filter 66, 76
 - Database functions 81
 - Date functions 80
 - Edit form 78
 - Examples 86
 - Functions 80
 - Misc functions 84
 - Numeric functions 82
 - Operators 79
 - Schematic functions 84
 - String functions 83
 - Tutorial 38
- filter button 66
- Filters
 - How do I 46
- FO templates 128
- Folders
 - Administration 302
- FOTemplate (administration) 292

- G -

- Gather data
 - How do I 46
- GET (administration) 349
- GIS 142
- GPS location 142

- H -

- Hardcopy 103
- Help 6
 - Tutorial 11
- Help system (administration) 193
- HelpReference (administration) 252
- High crash location list (administration) 337
- High crash locations 112
- Home page 90
 - Tutorial 10
- Hot spot list 112

- I -

- IcmRA_Admin (administration) 365
- IcmRA_Login (administration) 385
- IIS_WPG
 - Installation 149

Image 103
 Indexes (administration) 191
 Install
 Group administrator 172
 Installation (administration) 148
 Installing Crash Magic
 Administration 147
 Interactive login
 Administration 194
 Intersection
 Aliases 59
 Intersection list 112
 Intersection study type 141
 Intersections
 Duplicate 61
 IUSR_<machine>
 Installation 149
 IWAM_<machine>
 Installation 149

- L -

Label 100
 Landscape 128
 Landscape template 133, 134, 135, 136
 Launching Crash Magic
 How do I 44
 Layout 128
 settings 123
 Layout elements
 Chart 99
 Crash listing 108
 Diagram 103
 Layout 125
 Location list 120
 LayoutHelpers (administration) 261
 Layouts 121
 Layout elements 125
 LDAP
 Administration 196
 Legend 100
 List
 Tutorial 32
 Listing 106
 Overview 103
 Load template 66, 143
 Tutorial 36
 load template button 66

Location list 112
 Layout elements 120
 Log in
 Tutorial 9
 Log out
 Tutorial 42
 Login 87
 Active Directory (administration) 196
 Administration 194
 Database passthrough (administration) 195
 Form 87
 How do I 44
 Installation 173
 Interactive (administration) 194
 Magic Auto (administration) 195
 Shortcut 87
 Login: MagicAuto (administration) 350
 Lookup 301

- M -

Magic Auto login
 Administration 195
 MagicAuto
 Administration 349
 Mid-block schematic 102
 Misc functions 84
 multiple,instance,registry 149

- N -

Node
 Aliases 61
 Node list 112
 Node study type 141
 Numeric functions 82

- O -

Object symbols (administration) 269
 Online help 6
 Other symbols (administration) 270
 Output 121

- P -

Pan 66

Password 87
 How do I 44
 Tutorial 9
 Passwords (administration) 386
 Paste 66
 Paste and copy
 Tutorial 35
 paste button 66
 PDF 121, 128
 Phone number 5
 Portrait 128
 POST (administration) 349
 Pre-installation (administration) 148
 Print 66, 121
 Tutorial 41
 print button 66
 Project home page
 Tutorial 10
 Project settings 91
 Projects 90
 Home page 90

- Q -

Queries
 How do I 46
 Query 76

- R -

ReEntry
 Administration 349
 Refresh 66
 Remote Access
 Administration 363
 Reporting
 How do I 46
 Reports 94
 Charts 96
 Collision diagrams 99
 Crash listing 103
 High crash location list 112
 REST
 Administration 363
 Role (configuration creation) 279
 Round-about schematic 102
 Route milepost study type 142

- S -

Save 66
 Save template 66, 145
 Tutorial 36
 save template button 66
 Scanned report 103
 Schematic 100
 Schematic (administration) 248
 Schematic functions 84
 Schematics 102
 Tutorial 16
 Settings
 Chart 98
 Crash listing 105, 106
 Diagram 100
 Diagram (tutorial) 23
 High crash locations 115
 Layout 123
 Project 91
 Study 137
 shared user 306
 Shortcut
 How do I 44
 Side by side 128
 Side by side template 134, 135, 136
 SOAP
 Administration 363
 SOAP (administration) 391
 Spacing 100
 Split locations 61
 Starting Crash Magic 87
 Starting the program
 How do I 44
 Street name
 Aliases 50
 Streets
 Duplicate 61
 String functions 83
 Studies 136
 How do I 46
 Study
 Export 75
 Tutorial 12
 Study settings 137
 Study types
 Address 140

Study types

- All data 140
- Case id 141
- Intersection 141
- Node 141
- Route milepost 142
- XY rectangle 142

- Symbols: Object (administration) 269
- Symbols: Other (administration) 270
- Symbols: Vehicle (administration) 270
- SYS 307

- T -

- T schematic 102
- Table definitions (administration) 185
- Technical support 5
- Telephone number 5
- Template save and load
 - Tutorial 36
- Templates 128, 143
 - Load 143
 - Load and save 66
 - Save 145
- Test: MagicAuto (administration) 351
- Text labels
 - Tutorial 18
- Title 100
- Toolbar 66
- Troubleshooting (administration) 314
- Tutorial 9

- U -

- URL (administration) 349
- User
 - Login 87
- User (add)
 - Installation 180
- User group
 - Administration 305
 - Login 87
 - Tutorial 9

- V -

- Vehicle symbols (administration) 270

- W -

- WSDL (administration) 391

- X -

- XSL (administration) 294
- XSLT 128
- XY rectangle study type 142

- Z -

- Zoom 66

Endnotes 2... (after index)

Back Cover