Coplogic (DORS) Interface

OVERVIEW

The Coplogic web-based service, DeskOfficer Online Reporting System (DORS), allows the public to report incidents to the police without requiring an officer to fill out the first incident report.

The DORS system primarily covers non-emergency incidents without an initial suspect, such as reports of lost property, theft, identity theft, hit and run, and vehicle burglaries.

Upon submission, Coplogic issues a police report case number to the user and an e-mail is sent to the citizen with a printable .PDF attachment of the report that is suitable for insurance claims.

This number will be traceable in the online system as well as the agency's DataTrak Records Management System.

This interface to DORS through the LogiSYSAPI enables import of these incidents as cases into RMS. Once inside DataTrak, the case can then be assigned and edited as necessary by authorized RMS users.

Required Interface Actions:
1. Obtain case numbers from RMS;
2. Transfer new case data entered in DORS to DataTrak;
3. Allow addition of supplementals.

Interface(s): Coplogic
Vendor Tag: CPL

Direct access via the API is restricted to the following calls with this interface:

LS_InitialzeApi
LS_UpdateRecordVendor

See the **Function Calls** section for more details.
In the event of certain errors, system response results from two error-checking functions, LS_GetOpt and LS_StrError. Details are available in the **Configuration and Error-checking Functions** section.
See the **Error Reporting Table** for the list of the possible API error messages and their resolutions.
See the **.DEF File Glossary and Format Guide** description of the message request terms.

Exporting DORS Data

1. Initializing the API

For creation and update transactions via DEF files, vendors first access the LogiSYSAPI, calling the VendorAPI.dll using function LS_InitialzeApi with a valid username, password and a Vendor Tag.

2. Creating a New Record

To send the data to RMS, the vendor calls the VendorAPI.dll function LS_UpdateRecordVendor. This creates a new record in the WINRMSDB database, defined by the contents of .DEF file and a new record number is returned.

Minimum Required Fields for a Case

- **To complete a case record in DataTrak, data for the following fields <u>must</u> be supplied:**

Table	Mapped	Field	Source
Case	Case.CaseNumber	CaseNumber	(Supplied by API via function)
Case	Case.CaseAgencyName	CaseAgencyName	Supplied in DEF by vendor
CaseReport	CaseReport.RepCaseNo	RepCaseNo	Supplied in DEF by vendor
CaseReport	CaseReport.RepRepNam	RepRepNam = 'IR'	Supplied in DEF by vendor
CaseReport	CaseReport.RepOfficerID	RepOfficerID	Supplied in DEF by vendor
CaseReport	CaseReport.RepOffBadgeNo	RepOffBadgeNo	Supplied in DEF by vendor
CaseReport	CaseReport.RepOfficerName	RepOfficerName	Supplied in DEF by vendor

Other Formatting Guidelines and Requirements

- Timestamp fields must also either be filled with data, or set to '9999-12-31-24.00.000000' and cannot be left as NULL.

- The DataTrak system defaults all timestamp and date fields to the correct empty values, but the API requires that they be supplied when creating a case.

- This means that **Case.CaseBeginDate**, **CaseReport.RepDate** and **CaseReport.RepTime** fields cannot be NULL, either. This is true whether the function used creates empty cases or full cases.

- In addition, in order to make a CaseNumber in DataTrak, an Agency must be supplied for every new entry.

- CaseVehicle entries must supply a VehClass value.

- An Incident Description from Coplogic will be treated as a Narrative in the RMS system and is limited to 2000 characters.

- Character data supplied that is longer than the lengths documented for fields in the data mapping will be truncated.

- Fields for which there is no equivalent in our system, such as e-mail, will be mapped to the narrative.

- The codes for involved persons in the initial Coplogic reports map to the CaseGenericPerson.GenPerType as follows:

  C for Complainant
  S for Suspect
  W for Witness
  P for Additional Person

- Calls from Coplogic will be sent under a single designated user related to an officer within the RMS system. This user must be specified by Coplogic in the **CaseReport.RepOfficerID**, **CaseReport.RepOffBadgeNo**, **CaseReport.RepOfficerName** fields.

- Location Type and LocationType codes from Coplogic will be entered into the RMS system. This is the **CaseOffense.OffLocCode** field.

- AddrType values from Coplogic are mapped to the **CaseGenPerAddr.AddrType** field and can include:

  C for Cell Phone
  F for FAX
  H for Home
  O for Other/Generic Property
  P for Pager
  W for Work

- Incoming Property Class data from Coplogic will be mapped into the RMS CaseProperty.PropClass values. The PropFormType is the description of the PropClass code from site validation tables. When the Property Class data does not match a LogiSYS class, the value will be 'O' or 'Generic Property.'

- Formatting of CLOB data: ! followed by field name, followed by 10 digit length of text, followed by text.

The LogiSYS API Service

The LogiSYS API (Application Program Interface**)** service accepts properly formed requests from an external third-party program running on a remote machine, or connected via radio communications.

The LogiSYS API service executable is able to handle multiple requests from more than one program simultaneously and to route the request results directly to LogiSYS programs, or through the VendorAPI.dll, to the requesting external program.

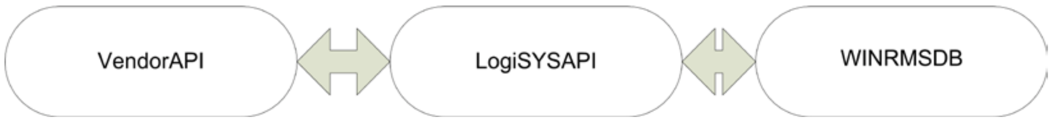
Third-party interfaces access the API service through the **VendorAPI.dll**.

This requires a Vendor Tag, used as an access key. This identifier is assigned by LogiSYS to identify and individualize vendor requirements for communications with the API.

Upon receipt of properly-formed requests, the API will then execute these requests by fetching, inserting or modifying data in the RMS Database (WINRMSDB) and return the results of the execution to the requesting program.

VendorAPI.dll

The VendorAPI.dll communicates function calls **to** the LogiSYS API service and returns data to the external program that executed the request.



Exported Functions of the API through the VendorAPI.dll

All functions and definitions are prefaced with "LS_" (LogiSYS) for quick recognition and to eliminate the possibility of a name-space conflict.

	Exported Functions of the API
1*	LS_InitialzeApi
2	LS_UpdateRecordVendor
	Configuration and Error-checking Functions
	LS_GetOpt
	LS_StrError

***Note the spelling of LS_InitialzeApi. This is correct for the application.**

Function Calls to the API through the VendorAPI.dll

The following describes how the functions are called by the Coplogic interface:

	Name	Usage
	LS_InitialzeApi	**Comments:** LS_InitialzeApi must always be called before any other functions in the VendorAPI.dll. **Message Text:** int LS_InitialzeApi(char *Username, char *Password, char *Tag) **Message Arg1:** *Username - The User Name for connecting to the database. **Message Arg2:** *Password - The password needed to connect to the database. **Message Arg3: ***Tag - The vendor tag, which is unique for each vendor. This is similar in function to a key, granting access to the API DLL. **Message Return Success:** LS_SUCCESS **Message Return Failure:** LS_ECONFIG - Configuration file provided by LogiSYS not found or is invalid. **Message Return Failure:** LS_EINVAL - Tag not recognized as a valid vendor ID. **Message Return Failure:** LS_UNKNOWN_ERROR - An unknown error occurred. Other more specific Error codes are returned by HandleError function. ***Note the spelling of LS_InitialzeApi. This is correct for the application.**
	LS_UpdateRecordVendor	**Comments:** This function will upload a record and return information on the affected records in the WINRMSDB. **Message Text:** __declspec(dllexport) int LS_UpdateRecordVendor(char *DefFile, char * char *ReturnFile); **Message Argument 1:** char *DefFile, Def File to be uploaded.

		Message Argument 2: char *ReturnFile, File containing a delimited list of records affected by Upload. **Message Return Success:** 0 for Success. All other values indicate an error. **Message Return Failure:** LS_ESEQERR - InitialzeApi has not been called. **Message Return Failure:** LS_ENOSERV - Server was not available. **Message Return Failure:** LS_EINVAL - TableName was not a recognized type. **Message Return Failure:** LS_ESERVER - An error occurred at the server.

LS_UpdateRecordVendor Functionality

LS_UpdateRecordVendor fills ReturnFile with the record corresponding to the .DEF file input.

The following is an example of a .DEF file:

- Any hard returns present in the examples have been added for ease of reading and are not required in the actual .DEF file.

- The FIELDDELIMITER and RECORDDELIMITER in the examples use printable characters for illustrative purposes only. The values for record and unit separators are specified by the DEF file creator on the first two lines. Binary values like '0x1C', '0x1D', '0x1E' and '0x1F' are recommended.

```
FIELDDELIMITER, ^ RECORDDELIMITER, @ CASE,A,CASENUMBER=IFC1^
CASEAGENCYNAME= ANYTOWN POLICE^CASEBEGINDATE=2010-05-14 12:01:01.000000^
CASEENDDATE=2010-05-14 12:02:01.000000^CASEADDR=1234^CITIZEN ST CASESTATE=MT^
CASEAPT=2A@CASEREPORT,A,REPCASENO=IFC1^REPAGENCYNAME= ANYTOWN
POLICE^          REPREPNAM=IR^ REPDATE=2010-05-14 12:12:12^ REPSTATUS=I^
REPTITLE=COPLOGIC CREATE CASE FULL TEST^REPTIME=09:09:09^REPOFFICERID=007^
REPOFFICERNAME=SMITH, JOHN JAMES  ^REPOFFBADGENO=XXX^
REPNARRINDEX=-3 @NARRATIVES,A,NARRKEYNO=-3^!NARRATIVE=000000000020coplogic
description @CASEOFFENSE,A,OFFCASENO=IFC1^OFFAGENCYNAME= ANYTOWN POLICE^
OFFREPNAM=IR   ^ OFFSEQNO=-1^ OFFCode=4000^ OFFType=B ^OffLarcenyType=01^
OFFMethod=01 ^OffEntry2=1^ OffPointOfEntry=1 ^OffLeftScene=1 ^OffPremCode=08^
@CASEGENERICPERSON,A,CASENO=IFC1^ AGENCYNAME= ANYTOWN POLICE^
REPNAME=IR ^GENPERSEQNO=-1 ^GENPERTYPE=C ^ETHNICITY=H ^RESSTATUS=C^
EYES=BLU^HAIRCOLOR=BLN^ AGE=21 ^HGT=69 ^WGT=200 ^BUSINESS=KINKOS^
@CASEGENPERNAME,A,CASENO=IFC1 ^AGENCYNAME= ANYTOWN POLICE
^REPNAME=IR^ GENPERSEQNO=-1 ^NAMESEQNO=-1 ^NAMETYPE=L^
LASTNAME=COPLOGIC_LAST^ FIRSTNAME=COPLOGIC_FIRST ^
MIDDLENAME=COPLOGIC _MIDDLE^ DOB=1968-10-21 ^DLNO=DL1234567890
^DLSTATE=ID^ SSN=012-01-0123^ SEX=M ^RACE=W @CASEGENPERADDR,A,CASENO=IFC1^
```

AGENCYNAME= ANYTOWN POLICE^ REPNAME=IR^   GENPERSEQNO=-1 ^ADDRSEQNO=-1^ ADDRTYPE=H   ^ ADDRADDRESS=30770 ^STAGECOACH BLVD ^ADDRSTATE=ID^ ADDRPHONE=4065551234 @CASEGENPERADDR,A,CASENO=IFC1^ AGENCYNAME= ANYTOWN POLICE^ REPNAME=IR ^ GENPERSEQNO=-1 ^ADDRSEQNO=-2 ^ADDRTYPE=W^ ADDRADDRESS=1010 RYMAN^ ADDRSTATE=1010 ^RYMAN ^ADDRPHONE=405 5551234^ @CASEGENPERADDR,A,CASENO=IFC1 ^AGENCYNAME= ANYTOWN POLICE ^REPNAME=IR ^ GENPERSEQNO=-1 ^ADDRSEQNO=-3 ^ADDRTYPE=C ^ ADDRPHONE=4065551AA@CASEPROPERTY,A,PROPCASENO=IFC1^ PROPAGENCYNAME= ANYTOWN POLICE ^PROPREPNAME=IR   ^ PROPSEQNO=-1 ^PROPCLASS=B ^PROPQTY=1 ^PROPBRANDNAME=B001^ PROPMODELNO=COPLOGICMODEL^ PROPSERIALNO=COPLOGICSERIALNO^ PROPOWNAPP=OAN^ PROPCOLOR=B^ PROPVALUE=100 ^PROPRECOVERED=1 ^PROPDESCRIPTION=COPLOGIC DESCRIPTION OF PROPERTY @CASEVEHICLE,A,VEHCASENO=IFC1^ VEHAGENCYNAME= ANYTOWN POLICE ^VEHREPNAME=IR ^VEHSEQNO=-1 ^VEHCLASS=A ^VEHLICTAGTYPE=PC^ VEHLIC=COPLOGIC_LICENSE^ VEHLICSTATE=ID^ VEHLICYEAR=2010^ VEHVIN=COPLOGIC_VIN^ VEHMAKESTR=FORD ^VEHMODELSTR=CROWN VICTORIA^ VEHTCOLOR=BLU ^VEHSTYLE=A ^VEHYEAR=2010 ^VALUESTOLEN=20000^ VALUERECOVERED=2000 ^VEHDESC=COPLOGIC VEHICLE DESCRIPTION@

Adding Supplementals

- **New Supplemental values should be 20001 or above.**

Supplemental reports are created by submitting a DEF file with a new CaseReport record. These records have the same requirements as the initial CaseReport submitted with the original Case DEF file, except that REPREPNAM values should be 20001.

The RMS-assigned numbers are listed in the file specified by the return file argument of the **LS_UpdateRecordVendor** function.

A REPCASENO must be this RMS-assigned case number and must be exactly 21 characters long, prepended by 0's (zeroes).

RMS will replace this number with the final, permanent value when the record is saved to the database.

- **A new DEF file must be submitted for each Supplemental.**

- **The order of creating supplemental records is important. The supplemental report must precede any reference to it.**

For example, a DEF file like the following <u>would</u> work:

FIELDDELIMITER,^
RECORDDELIMITER,@

CASEREPORT,A,REPCASENO=0000000000000QU100006^REPAGENCYNAME= ANYTOWN POLICE ^REPREPNAM=20001^REPOFFICERID=67   ^REPDATE=2010-05-15-.53.48.000000 ^REPOFFICERNAME=BOND, JAMES ORENTHAL   ^REPOFFBADGENO=XXX ^REPNARRINDEX=-3@
CASEGENERICPERSON,A,CASENO=0000000000000QU100006^AGENCYNAME= ANYTOWN POLICE   ^REPNAME=20001^GENPERSEQNO=-1^GENPERTYPE=C   @

While the next would fail because the report name referred to for CASEGENERICPERSON does not then exist:

FIELDDELIMITER,^
RECORDDELIMITER,@
CASEGENERICPERSON,A,CASENO=0000000000000QU100006^AGENCYNAME= ANYTOWN POLICE   ^REPNAME=20001^GENPERSEQNO=-1^GENPERTYPE=C   @
CASEREPORT,A,REPCASENO=0000000000000QU100006^REPAGENCYNAME= ANYTOWN POLICE ^REPREPNAM=20001^REPOFFICERID=67   ^REPDATE=2010-05-15-.53.48.000000^REPOFFICERNAME=BOND, JAMES ORENTHAL   ^REPOFFBADGENO=XXX ^REPNARRINDEX=-3@

DEF File Sample for a Supplemental:

FIELDDELIMITER, ^
RECORDDELIMITER, @
CASEREPORT,A,REPCASENO=0000000000000AN100291^REPAGENCYNAME=ANYTOWN^ POLICE REPREPNAM=20001^REPDIVISION=^REPOFFICERID=67^REPOFFICERNAME=TEST OFFICER NAME^REPOFFBADGENO=67^REPSTATUS=I^REPDATE=2010-09-14 21:33:53.000000^REPTIME=21:33:53^REPORTTIME=213353^REPNARRINDEX=-6^REPCREATEOFFID=OFF2^REPCREATEOFFNAME=OFFICER2^REPCREATEOFFBDGNO=222 ^ ENTERDATE=2010-09-14^REPTITLE=SUPPLEMENTAL 1@NARRATIVES,A,NARRKEYNO=-6 ^NARRDATE=9999-12-31 24:00:00.000000^NARRBLOBTYPE=^NARRDESCRIPTION=^ LOCALUPDATETYPE=^!NARRATIVE=000000000017test supplemental @

Configuration and Error-checking Functions

LS_GetOpt

Comments: Usually issued immediately after the **LS_InitialzeApi** function, **LS_GetOpt** checks against DataTrak's configurable option settings, providing the mechanism for getting VendorAPI.dll's options. Though not necessary for operation of the VendorAPI module, it is provided as a convenience for programmers. The only options currently defined or required are those that designate the API service port number and machine name.

Message Text: int LS_GetOpt

Message Arg1: *<Option Name> Current Valid Values, "EIEIOMACHINENAME", "EIEIOPORTNUMBER" or any vendor-defined option.

Message Return Success: Returns configuration information as a string representing option value.

Message Return Failure: LS_ECONFIG (-1010) /* Configuration file not found or invalid */

LS_StrError

Comments: Used along with LS_GetOpt and in conjunction with the VendorAPI.dll module or test program. LS_StrError provides descriptions for numeric errors returned from functions in the VendorAPI.dll. It is intended to assist in debugging problems with the VendorAPI.dll module. For a complete listing of the possible errors and their resolutions, see the Error Reporting Table.

Message Text: int LS_StrError

Message Arg1: Any Error Number returned from a LogiSYS API function.

Message Return Success: A string that describes the nature of the error.

Message Return Failure: LS_EINVAL (-1006) /* Argument or value provided is not valid */

Error Reporting Table

Constant Definitions	Action Required
// LOGISYS ERRORS (Messages limited to 2048 characters)	
#define LS_SUCCESS (0) /* SUCCESS */	None. LogiSYS accepted the DEF file and created the record(s) in the database.
#define LS_ESEQERR (-1001) /* Functions called out of sequence */	This indicates a programming error. You must call LS_InitialzeApi before calling any other LS function.
#define LS_EBADF (-1002) /* File operation failed */	Ensure the DEF file specified by DefFileName exists and is readable.
#define LS_EPERM (-1003) /* Permission denied */	Ensure that the file specified by FailedRecordsFile exists and is writable, or that the directory path that will contain that file exists and is writable.
#define LS_EBADDEF (-1004) /* DEF File is in an improper format */	This indicates the DEF has the incorrect format. The specified DEF file may have the wrong extension, or is missing delimiters in the text.
#define LS_ENOSERV (-1005) /* Server was not available */	The host or the LogiSYSAPI service itself is unavailable and LS_InitializeAPI must be called again.
#define LS_EINVAL (-1006) /* Argument or value provided is not valid */	The vendor tag specified is not valid. You must resend the DEF file after the error is corrected.
#define LS_ESERVER (-1007) /* An error occurred at the Server */	An error occurred at the LogiSYSAPI server. You must resend the DEF file after the error is corrected.
#define LS_ERROR (-1008) /* A Low-level support function failed */	An error occurred in low level VendorAPI.dll functions. You must resend the DEF file after the error is corrected.
#define LS_UNKNOWN_ERROR (-1009) /* Unknown error */	You will need to resend the DEF file once the error is diagnosed and resolved.
#define LS_ECONFIG (-1010) /* Configuration file not found or invalid */	The config file is invalid. You will need to resend this DEF file once the error is corrected.
#define LS_ENOTEMP (-1011) /* Unable to create temporary directory */	Could not create a temp file. Check file system permissions. You must resend this DEF file after the error is corrected.
#define LS_ESARGS (-1012) /* Arguments are invalid. */	Invalid arguments were passed to the API call. You must resend this DEF file after the error is corrected.
#define LS_UNSUPPORTED (-1014) /* Unsupported function for current Vendor */	A request type or function is not valid for the Vendor Tag used.
#define LS_RECORD_NOT_FOUND (-1015) /* Record doesn't exist in database */	A record specified in the request does not exist in the database. Reform the request with a valid record number.
#define LS_BAD_DEF_RECORD (-1016) /*Def Record refers to non-existent record */	The DEF file contains a reference to a record that was not included in the DEF file, but needs to be. You will need to resend this DEF file after the error is corrected.

.DEF File Glossary and Format Guide

Overview

The DataTrak Exchange Format (.DEF) is a file definition by which data can be transferred between applications and the RMS DB2 database.

This document describes the required format, tokens, and general requirements of the .DEF.

RS - The ASCII character 0x1E (Record Separator)
FS - The ASCII character 0x1C (Field Separator)
Record Type – Case only
Record List - A list of data elements, each of which includes a record type and a key list that uniquely identifies only one record of that type.

RecordType<FS><LogiSYS API Record Type><US>Key1<FS><Key value><US>Key2<FS><Key value><US>Key3<FS><Key value><US><RS>

LogiSYS API - This is the daemon that communicates between the external program and the RMS DB2 Database.

FIELDDELIMITER,<Field Delimiter>
RECORDDELIMITER,<Record Delimiter>

Example:

FIELDDELIMITER, ^
RECORDDELIMITER, @

(The FIELDDELIMITER and RECORDDELIMITER in the examples use printable characters for illustrative purposes only. Standard values for record and unit separators like '0x1C', '0x1D', '0x1E' and '0x1F' are recommended.)

Body

<TABLENAME>, <ACTION>, <FIELDNAME>=<FIELDVALUE>< FIELDDELIMITER> . . .
<FIELDNAME>=<FIELDVALUE>< FIELDDELIMITER> ...
<FIELDNAME>=<FIELDVALUE>< RECORDDELIMITER>

Specification for LOBData Columns:

The .DEF File format for columns of type CLOB or BLOB is:

!<FIELDNAME>=<FIELDVALUESIZE><FIELDVALUE><FIELDDELIMITER>

<FIELDVALUESIZE> is a 14 character 0-prepended number that gives the byte count of the <FIELDVALUE> data.

NOTE: The exclamation character is only valid if the <FIELDNAME> is a BLOB or CLOB.

<FIELDNAME> is the DB2 column name of the field. If the .DEF file contains LOB Data, the FIELDDELIMITER, and RECORDDELIMITER cannot be the exclamation character ('!').

Where <TABLENAME> is one of the legal RMS table names.

Where <ACTION> is one of: A
Specify Insert (A) of the record.

Fieldnames need only be specified for fields being added or modified. Any existing field values for the given record (if record exists) are untouched, unless specified.

Where VALUE (for example) = A String then 'String'

A Date then 'mm/dd/yyyy'
A Time then '01:01:01'
A Timestamp then '1988-12-25-17.12.30.000000'

Whitespace is ignored, except as part of a VALUE.

All RMS-designated "key" fields must be specified for each TABLENAME, regardless of ACTION type. The key field requirements for each TABLENAME are specified in the database.

VALUEs for fields that are RMS “code fields” must match the client-specified RMS validation table codes for that field. (The code from these tables, not the description, is saved to the RMS DB2 database). No checking at import time is performed. If the VALUE imported and saved to the RMS database is a non-valid client code, the DataTrak RMS GUI will display that value, but if the client clicks in the GUI field, the value will be "lost" as the legal values are displayed from the appropriate RMS validation table.

Import Program Functional Specification

Every key for a given TABLENAME must be included, with a value, in the import file record. Otherwise, the .DEF import reports an error.

When updating, if no record exists in the DB2 database with the given keys, the .DEF import reports an error.

Each TABLENAME has one key field that is the "unique control number" of that record, and often several key fields that hold the "unique control numbers" for the "parents" of that record. The .DEF import causes the generation of RMS-specific numbers for these fields, upon creation (Insertion) of a new TABLENAME record.

For example, multiple vehicles entered with a Case Report are assigned sequential control numbers, and each Case Report of a Case is assigned a unique control number.

- For Case records, this "unique control number" is the CaseNumber field.
- For CaseReport records, this "unique control number" is the RepRepName field.

- For CaseGenericPerson records, this "unique control number" is the GenPerSeqNo field.

And so on.

The .DEF import is responsible for knowing which of the key fields of a given TABLENAME is the "unique control number" field.

"Unique control numbers" at Insertion:

For a new record Insert ACTION of a TABLENAME record, the data values specified in the .DEF import file for these "unique control number" key fields must be "relative" control numbers.

This means that the numbers are relative and sequential for the record hierarchy of the records entered in this import file.

The relative control number for the top level parent module of Case must be prefaced by the characters "IFC" and followed by one or more numbers. The string can be up to 21 characters long.

The relative control numbers for child records must be a negative number.

For example, a third-party vendor specifying data for 1 (the initial) Case Report of each of 3 new Cases might number the Cases "IFC43", "IFC44" and "IFC45" and each Case Report "20001." The Case Report numbers can all be " -1" because Case Reports are children of Cases in the RMS record hierarchy, and the parent Case "unique control number" is also part of the key field of the Case Report.

Saving LOB Data

DEF files for tables with BLOB or CLOB columns use the same format to save data as other tables, but the field data for the LOB column has extra requirements:

1. The Field Name associated with the LOB data must be prepended with an '!'.
2. Following the Field Name and equal sign, there must be a 12-digit 0-prepended number that will be the length of the data in bytes of the LOB field.
3. Immediately following this number is the data for the LOB field.
4. The LOB field must be placed after the last field for the table and before the RECORDDELIMITER.
5. The referring column from the referring record must match the key number of the added LOB record.

Formatting of CLOB data: ! followed by field name, followed by 10 digit length of text, followed by text.

Example:

FIELDDELIMITER,^
RECORDDELIMITER,@

@NARRATIVES,A,NARRKEYNO=-
1^NARRDESCRIPTION=^!NARRATIVE=000000000015Hi there world!@

Requirements for an Add.

1. The Key Number associated with the table must be negative.
2. The record with LOB data must be after the record to which it is linked.
3. The referring column from the referring record needs to match key number of the added LOB record.

Mapping Tables – Coplogic Interface

INCIDENT

ID No.	Coplogic Field	WINRMSDB Data	Type	Size	Comments
1.1	**TEMPORARY NUMBER**	CASE.CASENUMBER	CHARACTER	21	IFC1,IFC2,IFC3, etc.
1.2		CASE.CASEAGENCYNAME	CHARACTER	39	
1.3	DATE	CASE.BEGINDATE	TIMESTAMP		
1.4	TIME OF OCCURRENCE		TIMESTAMP		Filled in from the time portion of the BeginDate timestamp.
1.5	DATE/TIME STARTED	CASE.CASEBEGINDATE	TIMESTAMP		
1.6	DATE/TIME ENDED	CASE.CASEENDDATE	TIMESTAMP		MUST be later than CaseBeginDate, or error will result.
1.7	REPORT FILED FROM	CASEREPORT.REPTITLE			Mapped to RepTitle. Example: “Coplogic Report from.”
1.8	LOCATION OF OCCURRENCE	CASE.CASEADDR	CHARACTER	49	
1.9		CASE.CASESTATE	CHARACTER	5	State is stored separately from the address.
1.10	APARTMENT #	CASE.CASEAPT	CHARACTER	5	
1.11	INCIDENT DESCRIPTION	NARRATIVES.NARRATIVE	CHARACTER	2000	
1.12		NARRATIVES.NARRKEYNO	INTEGER		Negative integer, must match CaseReport.RepNarrIndex.
1.13		CASEREPORT.REPCASENO	CHARACTER	21	IFC1,IFC2,IFC3, etc.
1.14		CASEREPORT.REPAGENCYNAME	CHARACTER	39	
1.15		CASEREPORT.REPREPNAM	CHARACTER	2	Set to 'IR' for the Initial Report. Set to unique number starting at 20001 for Supplementals.

ID No.	Coplogic Field	WINRMSDB Data	Type	Size	Comments
1.16		CASEREPORT.REPNARRINDEX	INTEGER		Negative integer, must match narratives.NarrKeyNo.
1.17	APPROVED BY	CASEREPORT.REPOFFICERID	CHARACTER	9	ID of the Coplogic user.
1.18	APPROVED BY	CASEREPORT.REPOFFICERNAME	CHARACTER	30	Name of the Coplogic user.
1.19	APPROVED BY	CASEREPORT.REPOFFBADGENO	CHARACTER	15	Badge number of the Coplogic user.
1.20	DATE	CASEREPORT.REPDATE	TIMESTAMP		
1.21		CASEREPORT.REPTIME	TIME		As there is no trigger for updating the RepTime from the RepDate timestamp, the RepTime value must be the same as the time portion for the RepDate timestamp.
1.22		CASEREPORT.REPSTATUS	CHARACTER	5	MUST Be 'I' or uneditable.
1.23	LOCATION TYPE	CASEOFFENSE.OFFLOCCODE	CHARACTER	5	Church, School and other codes.
1.24		CASEOFFENSE.OFFCASENO	CHARACTER	21	IFC1,IFC2,IFC3, etc.
1.25		CASEOFFENSE.OFFAGENCYNAME	CHARACTER	39	
1.26		CASEOFFENSE.OFFREPNAM	CHARACTER	2	Set to the value used in CASEREPORT.REPREPNAM.
1.27		CASEOFFENSE.OFFSEQNO	SMALLINT		Negative unique number per case.
1.28	INCIDENT CODE	CASEOFFENSE.OFFCODE	CHARACTER	25	Offense Code, the municipal code.
1.29	UCR CODE	CASEOFFENSE.OFFUCRCODE	CHARACTER	5	UCR Offense Code.
1.30	NCIC CODE	CASEOFFENSE.OFFNCICCODE	CHARACTER	10	NCIC Offense Code.
1.31	ATTEMPTED OR COMPLETED	CASEOFFENSE.OFFCSA	CHARACTER	5	Committed Suspected or Attempted.