Parasolid V35.1

Reporting Faults To Parasolid

January 2023

Important Note

Francis House
112 Hills Road
Cambridge CB2 1PH
UK
Tel: +44 (0)1223 371555
email: parasolid.support.plm@siemens.com
Web: www.parasolid.com

SIEMENS

Trademarks

Table of Contents

Introduction 1

1.1 Content of this manual

This manual explains how to generate and submit the data required to report a fault or make an enquiry to Parasolid Support for investigation. By following the guidelines in this document, you can create the best possible set of data to describe a given issue. This in turn can help speed up the time between reporting the issue and getting a PR (Problem Report) reference number for that issue; once you have been given a PR number, you can be assured that Parasolid development have all the information required to investigate the issue.

In addition, this manual explains how you can download the latest full and update (patch) releases of Parasolid for the versions and platforms you are including in your products.

The first priority for Parasolid Support is always to reproduce the problem you are reporting. If possible, reproducing the problem within the Parasolid test and debugging environment is preferable, since this is much more efficient for the subsequent identification and resolution of the issue within the Parasolid Development team.

- Chapter 2, “Data To Include In Fault Reports”, describes the data that you need to provide to Parasolid Support when reporting a fault or making an enquiry.
- Chapter 3, “How To Report A Fault”, describes how to send required data to Parasolid Support.
- Chapter 4, “Downloading Full and Patch Releases”, tells you where you can download Parasolid using an internet connection.
- Appendix A, “Updating The Frustrum”, describes how you can update legacy frustrum functions that were created prior to Parasolid V13.2 in order to take advantage of Parasolid’s native debug format.

1.2 Summary of fault reporting procedure

This section provides an overview of the steps typically required when reporting an issue to Parasolid Support. It does not cover all possible scenarios, and is intended only as a quick reference.

Step	Notes
Identify the failing function	If you are reporting a problem with Parasolid functionality, identify the PK or KI function call that is causing the problem.

Step	Notes
Gather relevant data	Generate debug information by wrapping calls to Parasolid debug functions around the failing PK or KI function. Optionally, create screenshots, mock-ups, result bodies, etc. to clarify any potentially ambiguous details and provide additional useful information.
Report the problem to Parasolid Support	If you have access to the GTAC online support service, create a new IR and upload all relevant data. Otherwise, contact Parasolid Support via email, and either attach relevant data or upload it to the secure FTP server.

For more detailed information about any of the steps above, refer to the rest of this manual.

1.3 Response from Parasolid support

Once a member of the Parasolid Support team has investigated your fault they will respond to you, quoting the Parasolid database reference number (e.g. IR1234567). You will also be contacted if the issue has been rejected (i.e. because Parasolid is behaving as designed, or the fault is caused by incorrect usage), or if more information is required to reproduce the issue. If you receive no response within a few days then please re-send the e-mail.

Data To Include In Fault Reports 2

2.1 Introduction

When reporting a fault (or illustrating an enquiry) the most convenient and useful data is normally debug data in the form of an XML file. This can be generated directly from a Parasolid session, using a format known as Debug XML: see Section 2.2, “Debug XML”, for instructions on how to do this.

In the vast majority of cases, Debug XML is sufficient for Parasolid Support to reproduce the problem. Sometimes, however, other supporting information is required or desirable for the speedy verification of the problem. Section 2.3, “Additional data”, describes the types of information you may need to provide.

In a small number of cases, Debug XML cannot be used to reproduce a problem. In these cases, you may be asked to provide other types of information instead. These are described in Section 2.4, “Other methods of reporting faults”.

Note: If you believe you have a problem implementing Foreign Geometry, please contact Parasolid Support before submitting fault data.

2.2 Debug XML

Debug XML is the standard mechanism for capturing all the information from a Parasolid API call that is required for Parasolid Support to reproduce the problem. Once the problem has been reproduced, this information is also integrated into the Parasolid test and debugging environment, to ensure that the problem never recurs in future releases of Parasolid.

The debug report file contains the following information:

- The version of Parasolid used to generate the file.
- The values of the current session parameters.
- All the PK function calls made, including their received and returned arguments
- The results of any PK functions.
- Entity arguments for functions, using node identifiers and a reference to a part – this allows the entity to be uniquely identified in a new session.
- Callback function data (optional: see Section 2.3.2, “Data for callback functions”, for information)
- Session information (optional: see Section 2.4.1, “Session files”, for information).
- XT parts, in either binary or text format (optional)

.

Note: The Debug XML system is not a true journaling system in that it is intended to capture exactly one modelling call. Debug XML files that contain multiple calls to Parasolid have a reduced chance of capturing useful information for anything other than the first call they contain.

2.2.1 How to generate Debug XML

You can generate Debug XML for a Parasolid API by wrapping the particular API function with calls that start and stop the debug reporting mechanism as follows:

```
PK_DEBUG_report_start ( key, options );
# Your API call
PK_DEBUG_report_stop ();
```

Note: The PK_DEBUG functions are declared in the header file `parasolid_debug.h` so you need an #include preprocessor directive for this file in order to compile calls to PK_DEBUG functions.

Here is a specific example for a boolean operation (where the input arguments for the boolean have already been set up):

```
PK_SESSION_set_journalling (PK_LOGICAL_false);
PK_DEBUG_report_start_o_t debugRep;
PK_DEBUG_report_start_o_m (debugRep);
PK_DEBUG_report_start("C:\\temp\\debug_xml.xml", &debugRep);
PK_BODY_boolean_2 (target, 1, &tool, &opts, &tracking, &results);
PK_DEBUG_report_stop();
```

PK_DEBUG_report _start receives the following arguments:

Argument	Description
`key`	A string which identifies the file to which the debug information will be written.
`options`	A set of options; It is not necessary to alter any of these except in specific circumstances (described throughout this chapter).

Journalling must be disabled for Debug XML to be generated: if journaling is enabled in your Parasolid session, you must turn it off by calling PK_SESSION_set_journalling before the call to PK_DEBUG_report_start, as shown in the example above.

Note: If your frustrum functions were written to integrate with a version of Parasolid earlier than V13.2, they may need to be modified to support Debug XML output. See Appendix A, “Updating The Frustrum”.

Warning: PK_DEBUG_report_start and PK_DEBUG_report_stop are debug functions and as such are intended for internal use only. **Products shipped to customers should not contain calls to either of these functions.**

2.2.2 How to verify Debug XML

A well-formed Debug XML file will open in a web browser and finish with the end tag:

```
</ParasolidFaultReport>
```

In some cases the XML may be truncated (particularly if there has been a run-time error or other unhandled error), but the output file is still useful to Parasolid Support. In many cases there is enough information in the XML to reproduce the problem anyway, such as the exact Parasolid build, basic platform information and session settings, input bodies, API details, arguments (including node IDs) and option structure contents, return structures and any error codes. You should ensure, therefore, that you submit any Debug XML that is generated from a Parasolid session to Parasolid Support, whether it is well-formed or not.

2.3 Additional data

In addition to Debug XML, other supporting information can often help illustrate a fault report or enquiry. Occasionally this extra information may be required, but in other cases it may still be very useful.

2.3.1 Data for regressions

In the case of regressions, it is important that data is provided to show both the “passing” and the “failing” cases; you need to provide evidence that the function being reported used to work or produced the desired result. Therefore when you report a regression, you should provide two Debug XML files, one for the passing case and one for the failing case. In some circumstances, result bodies may also be requested by Parasolid Support to help with the speedy resolution of the problem.

See also Section 2.3.4, “Other useful data”.

2.3.2 Data for callback functions

If the problem relates to a PK function that lets you specify a callback function definition, then Parasolid can generate debug XML for any calls to that callback function by setting the `report_cb` option to a non-default value.

Recording callback data in the debug XML is typically useful to Parasolid Support when they attempt to replicate the call made to the relevant PK function.

This option takes the following values:

Value	Description
PK_DEBUG_report_cb_no_c	Do not generate debug information for callback functions. The debug XML contains a pointer to the callback function itself, but no further information. This is the default.
PK_DEBUG_report_cb_function_c	Generate full debug information for any callback functions that are specific to a particular PK function. **Note:** Setting this value does not generate debug information for system-wide callbacks, such as those you register using PK_ATTDEF_register_cb.

2.3.3 Data for RTEs in Parasolid

As described in Section 2.2.2, “How to verify Debug XML”, encountering a run-time error in Parasolid may mean that the Debug XML is not well-formed. In such cases, you might be asked for additional information in order for Parasolid Support to reproduce it.

In particular, if the Debug XML generated is incomplete, and you know that the RTE applies specifically to setting a particular option value, then try generating complete Debug XML by setting the option to a different value. Submit this data to Parasolid Support, but make sure you clearly flag the option value that causes the RTE to Parasolid Support in your incident report.

> **Note:** Incomplete Debug XML can be created in circumstances other than an RTE (such as, for example, an infinite loop). In such cases, see Section 2.4, “Other methods of reporting faults”, for information.

See also Section 2.3.4, “Other useful data”.

2.3.4 Other useful data

In many cases, additional data can be useful for Parasolid Support to speedily verify your issue and to ensure that there is a common understanding of the problem. This section provides a non-exhaustive list of information that can sometimes be useful.

Type of issue	Notes
Platform specific, or SMP specific	If you suspect an issue is platform specific or specific to having Symmetric Multi-Processing enabled in Parasolid, then you should mention this in your report.
Unexpected or undesired results	If an operation produces unexpected or undesired results, the following information can be useful: ■ A screen shot clearly identifying the problem area. ■ A problem face highlighted using a different color. ■ A mock up of the expected result.
Different Parasolid versions producing different results	If two versions of Parasolid produce a different result, useful data could include: ■ The result bodies from each version. ■ A clear description of the differences where it is not visually apparent.

In all cases, when explaining what area of a part the problem occurs in, you should use Parasolid entity identifiers. An identifier is different from a Parasolid tag in that it is persistent across a save and restore of a model. You can find out the identifier of a given entity by passing the entity tag to PK_ENTITY_ask_identifier.

Sometimes, Parasolid Support may request further information, such as input or output bodies or details of preceding or subsequent operations. This is usually either to fully re-create the conditions that are required to reproduce the issue, to better understand the overall workflow which may be essential to determine the correct course of action, or to enable a more speedy resolution within Parasolid Development.

2.4 Other methods of reporting faults

In rare circumstances, Debug XML is not sufficient for Parasolid Support to reproduce the problem. Usually, these occasions are predictable and include:

- Transmit/receive issues.
- Problems that occur as a result of data already cached in a Parasolid session.
- Situations where Debug XML was not properly generated.
- Cases where it is necessary to replicate more than just a single call to Parasolid (typically in situations where a previous call has changed the Parasolid session in a way that needs to be replicated in order to reproduce the issue).

This section describes some of the information that can be useful to Parasolid Support in these situations.

2.4.1 Session files

Sometimes, a session file is required by Parasolid Support. This contains a complete snapshot of a Parasolid session, which can then be read into a new session to restore the internal state of the saved session: the contents of the internal data structures and caches will be exactly as they were before. In this way, certain issues that cannot be reproduced using a standard Debug XML file will become reproducible.

You can create session information from a call to PK_DEBUG_report_start by setting the `output_session` option to PK_DEBUG_output_session_bin_c. The resulting session data is included in the output file.

> **Note:** Most Parasolid APIs require that any entities specified as arguments are passed in using tags (the unique integer value assigned to all entities in a session). Although they are not persistent between sessions, tags are preserved when the session is saved, so you should record relevant tags immediately before the session file is generated, and include them in your problem report to Parasolid Support.

After examining this session data, Parasolid Support might subsequently ask you to produce similar information using PK_SESSION_transmit, though this is not normally required.

2.4.2 Body and API arguments

If the Debug XML is truncated (due to an RTE for example) then it is sometimes enough to submit the bodies and API details to Parasolid Support. Bodies should be provided in `x_t` or `x_b` format (i.e. using PK_PART_transmit), and saved just before the problem API call, together with the full set of arguments and options passed to the particular API.

If you supply data this way, refer to specific entities using identifiers, not tags. See Section 2.3.4, “Other useful data”, for more details.

2.4.3 Bodies and LISP

If Debug XML cannot be generated and you are familiar with the Parasolid KID environment, you can submit a LISP test that reproduces the problem. To ensure that the test is reproducible, use identifiers to refer to any entities: see Section 2.3.4, “Other useful data”.

We do not expect you to use this method unless you are already familiar with the Parasolid KID environment.

2.4.4 Reproducing in the application

If it is not possible for Parasolid Support to reproduce an issue using any of the methods already described, then it is likely that we will need to use your application directly.

The minimum information and data required to achieve this is:

- An appropriate copy of the application (and any installers etc.).
- The relevant application (part) files.
- Any licenses etc. required to install and run the software.
- Clear instructions for reproducing the issue.
- If required, instructions for attaching a suitable debugger to the application.

Note: A secure FTP account can be provided for any large file transfers that might be required in order to submit this information and data.

How To Report A Fault 3

3.1 Introduction

Once you have generated appropriate data, there are three principal ways that you can report a fault to Parasolid Support. In order of preference, they are:

- Use Siemens PLM Software's online support services. See Section 3.2.
- Send email to ps-support.plm@siemens.com. See Section 3.3.
- Contact us by telephone. See Section 3.5.

> **Note:** Although the telephone can be used for follow-up interaction with Parasolid Support, it is rarely possible to fully report an issue by telephone because of the need to submit data.

3.2 Online support services

Siemens PLM Software provides the following support services via the internet:

Support service	Description
Issue Reporting – Report new issues to Parasolid Support	The issue reporting mechanism lets you submit technical questions and fault reports online. Following submission of an issue, you are immediately given a tracking reference number for it. For a detailed description, see Section 3.2.3, "Issue reporting details".
Issue Status Tracking – Enquire status of existing issues	Once an issue has been submitted to Parasolid Support, you can inquire its status using the issue status tracking mechanism. For a detailed description, see Section 3.2.4, "Issue status tracking".
File Upload – Upload supporting data for new issues	You can use an FTP file upload facility to submit any data associated with an issue you have reported. Data uploaded to this server is only accessible by members of the Parasolid Support team, and cannot be viewed by other customers. For a detailed description, see Section 3.2.5, "File upload ".
Release Downloads – Download full and update releases of Parasolid	The release downloads mechanism allows you to download full and update (patch) releases of Parasolid. For a detailed description, see Chapter 4, "Downloading Full and Patch Releases".

All of these services can be accessed from the main menu page for Parasolid Support:

`https://www.siemens.com/sisw/parasolid-support/`

3.2.1 The WebKey mechanism

Access to each of these support services is controlled by an authentication scheme called “WebKey”. Your WebKey account username and password controls access to these services and information, and also protects the privacy of your data. By default, each customer is issued with a single Webkey account. Further Webkey accounts can be created for individual contacts on request: please contact Parasolid Support.

Note: WebKey accounts for Parasolid Customers are administered separately from those for other Siemens PLM Software customers. Any queries relating to your Webkey should be directed to Parasolid Support.

3.2.2 The GTAC support site

Parasolid’s online support services are provided through the GTAC (Global Technical Access Center) online interface. GTAC is the support facility utilised for the majority of Siemens PLM Software products. It is available at the following location:

`http://support.industrysoftware.automation.siemens.com/gtac.shtml`
The front page of the GTAC support site provides access to all of the Parasolid online support services. You can access each service from several different areas, as shown below:

Service	“Support” menu bar	“How Can We Help You?” menu	GTAC Featured Services menu
Issue Reporting	Call Handling > IR Creation	Call Handling > Customer Call Logging	Report an Issue
Issue Status Tracking	Call Handling > IR Status	Call Handling > Customer Call Tracking	Status of an Issue
File Upload	Downloads > FTP Server	Software Download > Files Upload	Download & Upload

When you select any of these services for the first time, you will be asked to log-in using your WebKey username and password.

Note: The Issue Reporting, Issue Status Tracking and File Upload services form only a subset of the services provided by GTAC, and there are other GTAC services that are not available to Parasolid Customers. Access to the services is controlled using the Webkey mechanism. If you believe that you have access to areas other than those described in this section, please inform Parasolid Support immediately for further guidance.

3.2.3 Issue reporting details

The service you use to log an issue online is also known as LOG-IR. When you select this service you are asked to enter either a server or a Sold-To ID. Enter whichever of these you are asked for, and click **Submit** to continue.

The LOG-IR screen is then displayed, for you to enter some details about the issue. The LOG-IR online help describes the various fields and the data that you need to enter. You must complete the “IR Problem Text” field, which should include the following information:

- The failing Parasolid function or problem area
- Source of fault report (for example, customer report or internal testing)
- The priority this issue has for you
- A clear description of what the problem appears to be and why it is a problem.

When you have finished entering the appropriate information, click **Submit**.. This creates an Incident Report (IR) and submits it to Parasolid Support for processing.

A confirmation screen is then displayed which includes the following information:

Information	Notes
A reference number for the IR.	Please make a note of this number and refer to it in any further discussions on the issue.
A link to the FTP upload area.	This is where you should upload any information associated with the IR to the FTP server. See Section 3.2.5, “File upload ”, for information on using the FTP upload area.
Links to other areas of the GTAC Support site.	These links provide easy access to creating other IRs, query the status of current issues, etc.

3.2.4 Issue status tracking

The service you use to query current issues is also known as QTAC. When you select this service you are asked to enter either a server or a Sold-To ID. Enter whichever of these you are asked for, and click **Submit** to continue.

The Call Viewer is then displayed. This offers four methods for searching for status information for your issues:

- Enter a specific issue number in the 'Call Number' box
- Search for issues using a range of criteria
- List all calls still open in the Parasolid Support team
- Access the Parasolid Release Download Site. See Chapter 4, “Downloading Full and Patch Releases”, for details.

Note: You may not be able to use this service for some legacy issues with reference numbers in the 4 million range, for example, IR4567890 or PR4567890. If you need to check the status of such issues contact Parasolid Support directly.

3.2.5 File upload

The File Upload service lets you submit any data associated with an issue to Parasolid Support. If accessing this service via the “Support” menu bar or the GTAC Featured Services menu, you will also need to click **Upload a File** on the subsequent screen.

Whenever possible, you should provide Parasolid data for an issue in Parasolid’s XML-based debug format. See Section 2.2, “Debug XML”, for information about how to generate this information.

Place all the files you are uploading into a single compressed file, and ensure that it is named appropriately. For example for IR1234567, the file should be named `IR1234567.7z`.

Note: It is your responsibility to apply the correct classification to your uploaded data to inform Siemens PLM if the data is non-exportable or under International Traffic in Arms Regulations (ITAR) restrictions.

If you are located outside of Europe, Parasolid Support will not be able to access your uploaded data and may not be able to investigate the IR unless you classify your files (via the GTAC interface) as both non-military and suitable for export. When you classify your data as such, it will be uploaded to a server located in Europe.

Upload the compressed file to the `/gtac` area, where it will be picked up by a member of the Parasolid Support team. Data uploaded to this area will only be viewable by members of the Parasolid Support team and not by other customers.

3.2.6 Automatic email notification

GTAC online services makes use of automatic email notifications to keep you informed about the progress of your issues. You will receive an automatic email in the following circumstances:

Event	Why the email is sent
IR Creation	When you first submit an issue via the online support services, or when Parasolid Support submits an issue on your behalf.
IR to PR Conversion	When an issue has been investigated and proves to be a fault in Parasolid that requires fixing, or requires further investigation by Parasolid Development for any other reason, the IR will be converted to a PR.
IR to ER Conversion	When an issue has been investigated and proves to be a function limitation in Parasolid requiring project work, the IR will be converted to an ER. Note that this does not indicate that the project has been scheduled.
PR Closure	When a PR associated with an IR is closed for whatever reason (e.g. fixed or rejected).

These automatic notifications are sent in addition to other email communications from Parasolid Support with regards to your issues.

By default, automatic emails are sent to the person that submitted the problem. For this to be possible, GTAC must either have a record of your email address, or you must enter it manually when you submit the issue online. If GTAC has no record of your email address, then the email is issued to the system manager contact we have recorded for your company. Typically, this may be the development manager for your organization.

3.3 Email

If you do not have access to GTAC, you can still submit issues via email.

For each problem or set of related problems, send a separate email to Parasolid Support `ps-support.plm@siemens.com` containing the information needed to reproduce the fault. The required information is detailed below.

Information	Notes
Email subject	The **subject** of the email should contain the following information, at least: ■ Your reference number ■ The failing Parasolid function or problem area ■ Source of fault report (e.g. customer report or internal testing) ■ Priority (see below) For example: **FR3564: PK_BODY_boolean_2 / Customer / Critical**. Using a meaningful subject allows Parasolid Support to prioritise problems without having to read each email in detail.
Email body	The **body** of the email should contain: ■ The debug information generated by Parasolid ■ A clear description of what the problem appears to be and why it is a problem for you The following information is also useful to Parasolid Support: ■ Your application name and version ■ A part file from your system (useful for verifying that the fix works in your product and sometimes helpful for our investigation)
Attachment	Either attach a single compressed file containing all your data, or upload your data using the FTP facilities described in Section 3.4, “Secure FTP facilities” **Note**: To alert Parasolid Support to the email attachment, please add the file name and file type in the covering email. As some compressed files may be blocked by our security network, we recommend that you use 7z format. There are limits on email attachment size: if you need to provide large amounts of data, we recommend using the FTP facilities. If you use the FTP facilities, you **must** include the name of the compressed file you uploaded in the body of your email.

Note: When emailing your data to Parasolid Support, it is your responsibility to ensure that the data is exportable to the United Kingdom and complies with regulations such as those laid out under International Traffic in Arms Regulations (ITAR).

3.4 Secure FTP facilities

If you need to provide large amounts of data, or you do not wish to attach data to an email, you can place the data on our FTP site.

The FTP site address is `ftp-cambridge.ugs.com`. You can use any FTP client you wish, so long as it has the ability to set up a connection using SFTP (Secure File Transfer Protocol). Clients such as Filezilla, CoreFTP and FireFTP all support this protocol. Log on to this server using the secure user name and password allocated to you.

There are multiple ways of setting up a connection using SFTP and the method differs from client to client. Some clients display SFTP as a protocol, others list it as a security option, while others let you specify SFTP in the address link by changing `ftp://` to `sftp://`.

If you are a UNIX user, and usually access the FTP via the command line, you need to log on to SFTP using `sftp yourloginname@ftp-cambridge.ugs.com`.

3.4.1 Adding FTP data

Create a single compressed file containing all your data. Please include a `readme.txt` in this compressed file that includes all the additional information detailed in your email. Place the file in the `/incoming` folder.

If you place data on the FTP site, you must send an email to Parasolid Support specifying the location of the files.

3.5 Telephone

If you wish, you can contact Parasolid Support via telephone on +44-(0)1223-371555. Please be aware that this is a standard UK number, and that our telephone lines are only manned during UK business hours, Monday to Friday.

Note: Please ignore any information included in generic GTAC documentation, which provides other contact details for technical assistance. You should use the above contact details for all issues relating to your use of online support services for Parasolid.

Downloading Full and Patch Releases

4

You can download both full and update (patch) releases of all current maintained versions of Parasolid from the Parasolid Release Download site. Available releases are organized by both Parasolid version and platform.

Other information on the page includes:

- Advance notification of future compiler and operating system version changes.
- Ability to download support tools such as the Parasolid Jumpstart Kit.
- Details on our current policy on the frequency of update releases, and which versions we are currently maintaining.

The Parasolid Release Download site can be accessed via the Parasolid Support main menu at

`https://parasolid-support.industrysoftware.automation.siemens.com/`

Note: The GTAC site includes a facility for release downloads, but the Parasolid system is separate from this.

Updating The Frustrum A

A.1 Introduction

If you use PK_DEBUG_report_start to create Debug XML with the `output_as_journal` option set to PK_LOGICAL_false (the default value) then the debug information is output to a new file with guise type FFCDBG. This guise type was introduced at Parasolid V13.2. If your frustrum implementation is older than this, you need to update a number of frustrum functions to support this guise type.

The following functions need to be modified or created, as indicated:

Function	Notes
`guise_string`	Modified
`filetype_guise_string`	Modified
`write_xml_header`	New
`OpenWriteFrustrumFile`	Modified
`WriteToFrustrumFile`	Modified

This appendix contains example source code for each of these functions.

Note: The following examples are based on the frustrum implementation used in the C++ Example Application contained in the file `frustrum.cpp`. The exact details for your frustrum may vary.

A.1.1 guise_string (modified)

Returns a pointer to a lowercase string which declares the file guise (that is rollback, snapshot, journal, transmit, schema, license and debug report).

```
static char* guise_string( int guise )
       {
       static char ffcrol[] = "rollback";
       static char ffcsnp[] = "snapshot";
       static char ffcjnl[] = "journal";
       static char ffcxmt[] = "transmit";
       static char ffcxmo[] = "old_transmit";
       static char ffcsch[] = "schema";
       static char ffclnc[] = "licence";
       static char ffcxmp[] = "transmit_partition";
       static char ffcxmd[] = "transmit_deltas";
       static char ffcdbg[] = "debug_report";
 switch ( guise )
        {
        case FFCROL: return ffcrol;
        case FFCSNP: return ffcsnp;
        case FFCJNL: return ffcjnl;
        case FFCXMT: return ffcxmt;
        case FFCSCH: return ffcsch;
        case FFCLNC: return ffclnc;
        case FFCXMP: return ffcxmp;
        case FFCXMD: return ffcxmd;
        case FFCDBG: return ffcdbg;
        }
        return g_unknown_value;
        }
```

A.1.2 filetype_guise_string (modified)

Returns a pointer to a filetype string for the specified guise.

```
static char* filetype_guise_string( int guise )
       {
       // Here we are assuming 3 character extensions
       static char ffcsnp[] = ".N";
       static char ffcjnl[] = ".J";
       static char ffcxmt[] = ".X";
       static char ffcsch[] = ".S";
       static char ffclnc[] = ".L";
       static char ffcxmo[] = ".XMT";
       static char ffcxmp[] = ".P";
       static char ffcxmd[] = ".D";
       static char ffcdbg[] = ".XML";
         switch( guise )
         {
         case FFCSNP:
         return ffcsnp;
         case FFCJNL:
         return ffcjnl;
         case FFCXMT:
         return ffcxmt;
         case FFCSCH:
         return ffcsch;
         case FFCLNC:
         return ffclnc;
         case FFCXMO:
         return ffcxmo;
         case FFCXMP:
         return ffcxmp;
         case FFCXMD:
         return ffcxmd;
         case FFCDBG:
         return ffcdbg;
         }
 return end_of_string_s;}
```

A.1.3 write_xml_header (new)

Writes the standard XML header to the file.

```
static void write_xml_header( file_p file_ptr, const char* pd2hdr, int pd2len,
         int* ifail)
         {
         char buffer[max_header_line];
         // <?xml version
         strcpy( buffer, "<?xml version=\"1.0\" ?>" );
         write_to_file( file_ptr, buffer, 1, strlen( buffer ), ifail );
         if ( *ifail != FR_no_errors ) return;
         }
```

A.1.4 OpenWriteFrustrumFile (modified)

Opens file to be written and writes to it the standard file header.

```
void OpenWriteFrustrumFile( const int* guise, const int* format,
                            const char* name,
         const int* namlen, const char* pd2hdr, const int *pd2len,
         int *strid, int *ifail )
         {
         char keyname[max_namelen+1]; /* holds key + null char */
         char filename[max_namelen+1]; /* holds key + extension */
         FILE *stream;
         file_p file_ptr;
 *ifail = FR_unspecified;
          *strid = null_strid;
 if (file_count == max_open_files)
           {
           *ifail = FR_open_fail;
           return;
           }
 strncpy( keyname, name, *namlen );
          keyname[*namlen] = end_of_string_c;
 strncpy( filename, name, *namlen );
          filename[*namlen] = end_of_string_c;
 if ( *guise == FFCSCH )
           {
           // add (and decode) a prefix to the filename
           extend_schema_filename(filename);
           }
 {
           /* add the file extension */
           char *gui = filetype_guise_string( *guise );
           strcat( filename, gui );
           if( *guise != FFCXMO && *guise != FFCDBG )
           {
           char *fmt = filetype_format_string( *format );
           strcat( filename, fmt );
           }
           }
 check_valid_filename( filename, ifail );
           if ( *ifail != FR_no_errors )
           {
           return;
           }
 /* open file for writing */
           stream = fopen( filename, g_fopen_mode_write );
           if (stream == 0 )
           {
           //*ifail = FR_already_exists;
           *ifail = FR_open_fail;
           return;
           }

/* Continued ... */
```

```
/* ... Continued */
new_open_file( stream, *guise, *format, write_access,
           filename, keyname, &file_ptr, ifail );
           if ( *ifail != FR_no_errors )
           {
           return;
           }

 if ( *guise == FFCDBG )
           write_xml_header( file_ptr, pd2hdr, *pd2len, ifail );
 else
           write_header( file_ptr, pd2hdr, *pd2len, ifail );
 if ( *ifail != FR_no_errors )
           {
           return;
           }
 *strid = file_ptr->strid;
           *ifail = FR_no_errors;}
```

A.1.5 WriteToFrustrumFile (modified)

Write buffer to open file

```
void WriteToFrustrumFile( const int* guise, const int* strid, const int* nchars,
                  const char* buffer, int* ifail)
   {
   file_p file_ptr;
   *ifail = FR_unspecified;
 /* find the file info for this stream-id */
          for ( file_ptr = open_files; file_ptr != NULL; file_ptr = file_ptr-
>next           )
          {
          if (file_ptr->strid == *strid) break;
          }
 if (file_ptr == NULL)
          {
          *ifail = FR_internal_error;
          return;
          }
          /* check file guise */
          if (*guise != file_ptr->guise)
          {
          *ifail = FR_unspecified;
          return;
          }
 /* check access */
          if (file_ptr->access != write_access &&
          file_ptr->access != read_write_access)
          {
          *ifail = FR_unspecified;
          return;
          }
          write_to_file( file_ptr, buffer, 0, *nchars, ifail );
          if ( *ifail != FR_no_errors )
          {
          return;
          }
 /* If we are writing a journal or debug report file then flush the
    buffer - this to ensure that in the event of crash as much data is preserved
    as possible */
 if (*guise == FFCJNL || *guise == FFCDBG)
          fflush( file_ptr->stream );
 *ifail = FR_no_errors;
```