Registered File Support for Critical Operations Files at SIRTF

G. Turek, T. Handley, J. Jacobson, J. Rector

Infrared Processing and Anslysis Center / SIRTF Science Center
California Institute of Technology, Pasadena, California 91125

Abstract. The SIRTF Science Center's (SSC) Science Operations System (SOS) has to contend with nearly one hundred critical operations files via comprehensive file management services. The management is accomplished via the registered file system (otherwise known as TFS) which manages these files in a registered file repository composed of a virtual file system accessible via a TFS server and a file registration database. The TFS server provides controlled, reliable, and secure file transfer and storage by registering all file transactions and meta-data in the file registration database. An API is provided for application programs to communicate with TFS servers and the repository. A command line client implementing this API has been developed as a client tool. This paper describes the architecture, current implementation, but more importantly, the evolution of these services based on evolving community use cases and emerging information system technology.

1. Introduction

The registered file system (TFS) is based on a client server architecture where clients are built upon the TFS application program interface, or API. Each server is connected to a Registered File System composed of a database, where meta-data is stored, and a file system where files managed by TFS are stored. The registered file system presents a virtual interface made up of a collection of file types. File operations (add files, get files, or add or get information about files) are conducted in the context of a file type. Files types have names, like "image", "ephemeris", or "cmdCdl", so one adds an "image" file or one gets an "ephemeris" file or one lists the "cmdCdl" files. The user does not need to know the files actual location or its server; the user just needs to know about file types. TFS is secure, not much may be done unless permission to execute commands for different file types has been granted.

2. TFS Architectural Overview

Figure 1 shows a federated TFS configuration with two TFS servers in an operational deployment, there can be many, but, here, two are shown for illustrative purposes. The system is federated for the two servers manage one or more file types in common, hence they must cooperate when they make changes. Having the servers write state information about file changes into the RFS database accomplishes this cooperation. The configuration as shown has a particular application. One TFS server supports user connections, while a second performs administration functions that require significant server support in their own right. File-database synchronization checking and file-integrity checking are the two administration functions that may require significant server resources so that running them on user-communities servers might affect performance. One or more TFS servers manage the file types defined in the Registered File System database. From a user's point-of-view, there is only one server for TFS service for the user does not work with servers, only with file types - the mapping between file types and servers is done in the TFS Domain file. If more servers are added, and the file types redistributed, the user only need be given a new Domain file that updated the file type to sever mapping. It is often a good idea to start with one or two servers and monitor the load on the servers. This performance information is then used to modify the configuration, adding more servers as required and redistributing the set of file types across servers so the best performance is achieved.

2.1. TFS Server

A TFS server has a number of input and output channels as illustrated in Fig. 2. I/O channel-related parameters for a TFS server are specified in the serverParameters table.

ffl User port: TCP port for new user connections
ffl Admin port: TCP port for new admin connections
ffl Event port: TCP port for out-of-band events
ffl Dynamically assigned client port: established by server's TCP protocol to service each client
ffl File System: servers can connect to multiple file systems RFS database: stores server state, configuration and file management information. Each TFS server has one or more continuously open connections to the RFS database
ffl Log file: each TFS server has an independent log file
ffl Significant events database: server errors that require immediate notice are logged in the sigEvents database.

2.2. TFS Clients

Client programs that communicate with TFS servers are built using the TFS user and administration application program interfaces, (APIs). The command line client, tfs, is a command line utility that allows interactive access to the TFS server. Users log onto the system, and then navigate file types using the ct (changeType) command. A number of operations are allowed depending on the user's permissions (read, write, delete). The command line client, tfsAdmin, is a command line utility that allows interactive access to the TFS server for administration purposes. Server, file system, user, and database administration functions are provided. In Figure 1, the tfsAdmin tool is connected to the user support server for monitoring purposes. Applications implementing the TFS API obtain connection information from the TFS Domain file. The domain file maps file types to servers, so it is essential that all applications and user

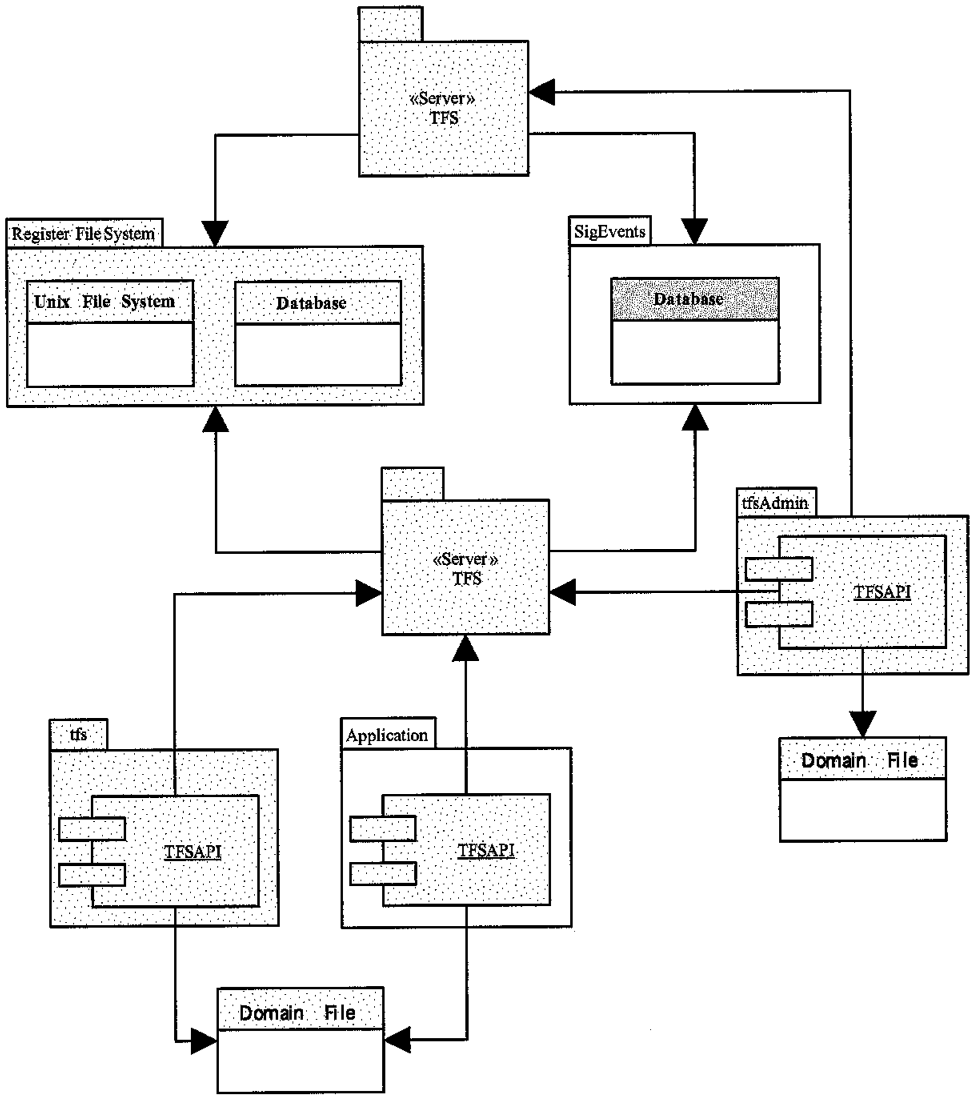

Figure 1. Federated TFS Configuration

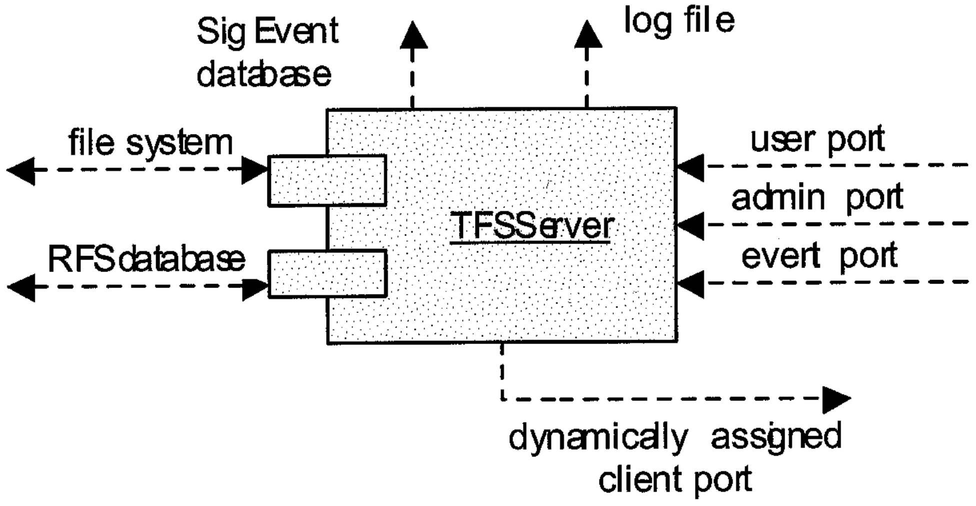

Figure 2. TFS Input and Output Channels

communities have a complete and up-to-date domain file. Since the contents of domain files are written in ASCII text, the updated files can be emailed to user communities. TFS domain files are generated from data held in the RFS database tables, the servers, and fileTypes being managed within the service.

2.3. Registered File System

The Registered File System consists of two major components: a file system and a database for storing data about the file system. TFS servers support files on any type of file system, local, remote, SAN based, NAS based. TFS servers may share files with other programs that know nothing about TFS; although it is best that these other programs only read copies of files that are locked in the file system. It is important to note that a locked file cannot be modified by any TFS server. Information about the TFS system and the files that it manages is stored in the Registered File System Database (RFS DB). Each file type described in the RFS table fileTypes has a file system directory associated with it, that is, the location of the files of the specified type.

2.4. System Requirements

All TFS Java software requires Java SDK 1.3 or later. Additionally, TFS servers require:

ffl Java(tm) 2 SDK, Enterprise Edition (1.2.1 or later)
ffl Java(tm) Secure Socket Extension (1.0.2 or later)
ffl Java Naming and Directory Interface(tm) (JNDI) File System Service Provider (1.2 beta 3 or later)
ffl C compiler for JNI functions
ffl File space for TFS distribution (2Mb) and server log files
ffl TFS account with rwx file access privileges on the file system used by files serviced, and access to the RFS database(s)

3. Implementation at SIRTF

Figure 3 shows the implementation of TFS at the SIRTF Science Center's (SSC). TFS will manage critical system operations files, data products and archival data as well as provide a file transfer mechanism between the SOS and the Mission Operations File System located at JPL.

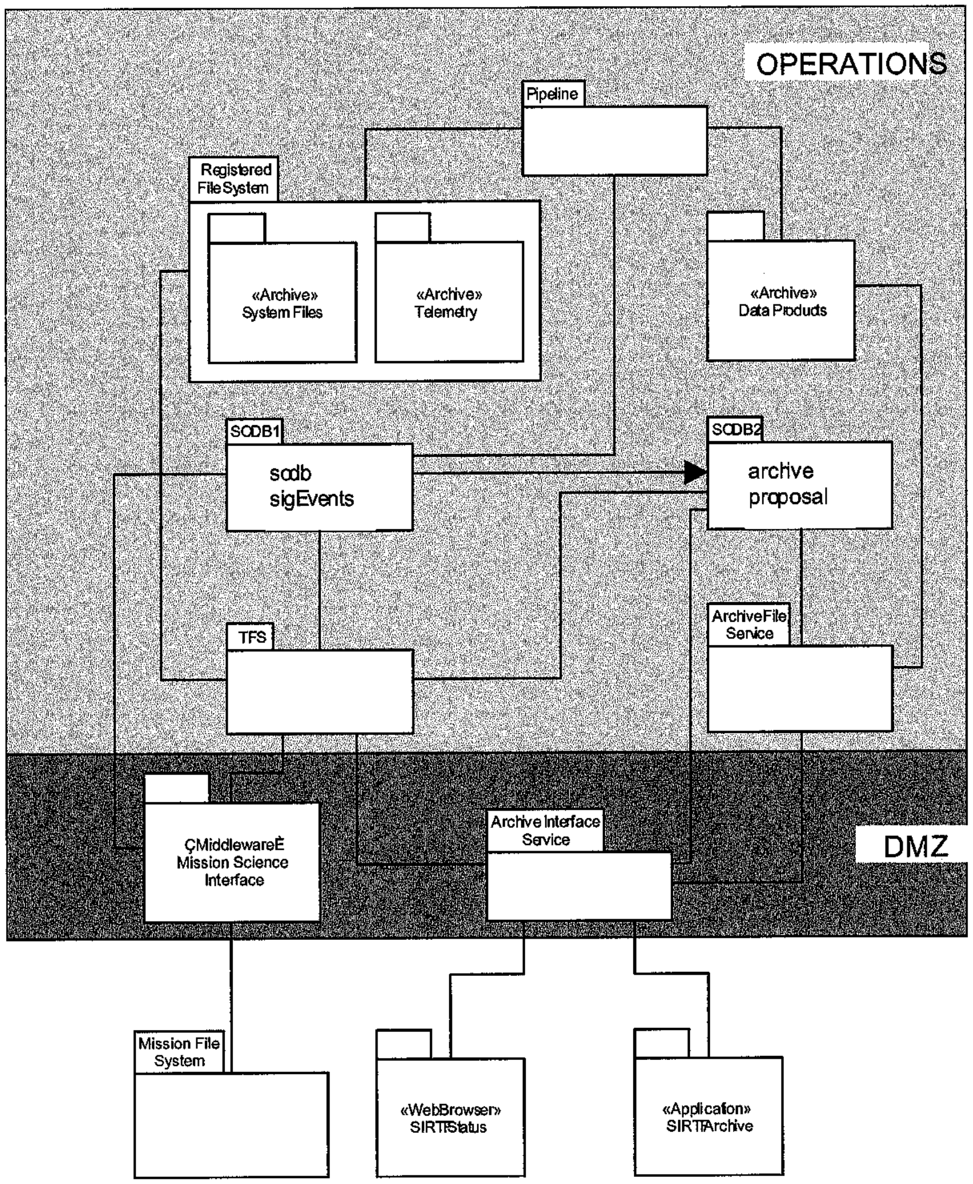

Figure 3. SIRTF Operations Configuration