

The T_EX Font Server Project

Version 1.0

Hippocrates Sendoukas

September 25, 1999

Contents

1	Introduction	2
2	Project Description	3
3	Server requirements (TFS Interface)	4
3.1	Query Mechanism	4
3.2	Response Format	4
3.3	Error Codes	5
3.4	Available Queries	5
4	Client requirements	8
5	Sample Server Implementation	8
6	Sample Client Implementation	8
7	Future Plans	9
8	Conclusion	9

1 Introduction

The setup of fonts in the \TeX world is a complicated and probably frustrating experience for almost all new (and even for many seasoned) \TeX users. Several factors contribute to this difficulty:

1. **Non-scalability of Metafont output:**

The “normal” \TeX fonts are usually produced by Metafont which is a very fine tool, but the produced fonts are bitmaps that cannot be scaled for resolutions other than the one for which they were produced. This means that the user has to obtain a series of files for each font according to the resolutions that he needs. The setup of Metafont is not that complex, but it’s not easy either; the added complexity brought on by the Metafont modes makes things even worse.

2. **Non-standard installation procedures for non- \TeX fonts:**

There are alternatives to Metafont, namely Adobe Type 1 and TrueType fonts. These fonts solve the scalability problem, but they introduce some complications of their own: their setup for most DVI drivers is harder than the setup of Metafont fonts, because most DVI drivers use their own method to deal with such “non-standard” (at least from a \TeX /Metafont perspective) fonts. One often needs multiple files for these fonts (eg., the actual font, one or two metrics files, an encoding file and a map file) and then one needs to manually enter this information in several places in order to enable the DVI drivers to use those fonts. This is clearly inconvenient and it has to be addressed if we want to simplify the usage of \TeX .

3. **Complications of Virtual Fonts:**

The introduction of the virtual fonts also added another dimension to the font puzzle. It introduced some more files to be considered and even more settings to adjust in the DVI drivers.

4. **Portability Problems:**

The DVI format is meant to be device-independent, but it is *not* meant to be portable. That is, I can process a given file with `dvips`, `xdvi` and `dviwin` and get practically the same results (assuming that I have set up all three drivers properly and the DVI file uses only specials that are acceptable by all drivers), but the situation is more complicated if I want to send the DVI file to another person; maybe that person uses the same DVI drivers but he does not have the same font setup that I have. This is a very common problem once the document uses any of the “non-standard” fonts and it significantly affects the easy interchange of DVI files.

5. **Difficult Design/Adaptation of non- \TeX fonts:**

The nature of \TeX imposes several constraints on the fonts that can be used; if one for example attempts to use a non- \TeX text font, one will find that the mathematical text does not match the text font. This is unacceptable, since \TeX is mostly used for technical material. There are some packages that simplify the

adaptation process, but the procedures are definitely involved and arduous for the non-experts (at least if one wants to use a non-Adobe font).

The T_EX Font Server (TFS) project is an attempt to simplify the use of fonts in T_EX-related software (DVI drivers in particular); it completely solves the first three problems and it has the potential to reduce or even solve the fourth problem (portability). The fifth problem (design or adaptation of new fonts) is not relevant from a DVI point of view; it has to be dealt by macro/package authors, so it is clearly beyond the scope of this project.

This document presents the project, describes its interfaces in detail and presents a sample implementation demonstrating a working TFS. The project is still in its infancy; all comments, ideas and any feedback are very welcome.

2 Project Description

The objective of the TFS project is to enable *any* user, no matter how inexperienced he may be about T_EX, to preview and print correctly *any* DVI document that uses freely available fonts. The basic idea of the project is to leverage the Web facilities to reduce or eliminate any font installation actions that need to be taken by the user of a DVI driver. In general terms, the installation of fonts requires some specific actions which have already been automated to some extent and for some DVI drivers, but this automation is neither complete (that is, some action is still required by the user), nor universal (the user's actions depend on the DVI driver that he is using).

The logic behind the TFS project is simple: whenever a DVI driver encounters a missing font, it will be able to consult a TFS server and find out the nature of the font (whether it is normal or virtual), as well as download the required files in whatever format is preferred by the DVI driver (or its user). Once those files are downloaded, the DVI driver can proceed normally; there will be no need for any input from the part of the user. The only requirement is that the computer executing the DVI driver has access to the Internet in order to communicate with the TFS server. One can also envision a similar scenario for an intranet; for example, an organization can set up a local TFS server fielding requests from local users; if a request cannot be satisfied by the local TFS server, the DVI driver can consult a "secondary" public TFS server; in this way, the users get a fast response from the local TFS, but also have access to more "exotic" fonts should the need ever arise.

One could implement the TFS server as a standalone server (eg., like a POP server), but this would be a waste of human resources; the simplest approach is to implement the TFS server as a CGI program running on top of a Web server. The CGI interface supplies the required functionality, is quite easy to program and test, and is available on practically all Web servers,

The intent of this project is to standardize the *interface* presented by the TFS server to its clients; the implementation of the server is a completely separate issue. We present a free sample implementation written in standard C which uses plain text files for its data; our particular choices were guided by our desire to make this implementation free and as portable as possible. Nothing prevents another implementation written

in another language or using another method to organize its data, from being called a “TFS server”, *as long as it follows the interface* presented in the following section.

3 Server requirements (TFS Interface)

3.1 Query Mechanism

The interface between the TFS server and its clients is quite simple; a client contacts the server and makes a query; the server responds to the query by consulting its database and returning the appropriate information. The entire conversation is performed is defined at the CGI level on top of the HTTP protocol. The TFS server *must* be able to handle all queries specified in this document; otherwise it cannot be considered a TFS server at all. The TFS queries take the form of a URL to a CGI program:

```
http://lyra/tfs?op=...&key1=value1&key2=value2...
```

where:

- “http://lyra/tfs” is the URL of the TFS server (the actual executable; *not* the Web site).
- “op=...” indicates the desired operation (query). Obviously, this is the most important CGI parameter. Each operation may also require some parameters; these are given by the following keys.
- “keyN=valueN” is a parameter pertaining to the query specified by the “op” key.

3.2 Response Format

There is a key parameter common to all TFS queries:

```
“fmt=...”
```

This parameter indicates the desired format of the server’s response. If this parameter is equal to the string “html”, then the server’s response will be in the HTML format; otherwise, the server’s response will be in a plain text format (which is also the default format in case this parameter is missing). Practically all DVI drivers will use the plain text format; the HTML format is only intended to provide an easy Web interface to the server, so that users can explore the server’s contents directly from their Web browsers. This is the interface used when you visit the web site of a particular TFS.

Since the HTML response is intended for humans, it is not subject to any strict syntax requirements. The only requirements are that it must conform to the HTML syntax and it should provide at least as much information as the plain text response.

The plain text response is intended to be read directly by DVI drivers and other TFS clients; as such, it must obey a rather strict syntax. Of course, the response varies with each type of query; however, the first line of each plain text response indicates the outcome of the query and it applies to all queries. This line should consist of a number

indicating an error code, followed by a colon (:) followed by a description of the error code (this description is meant to be presented to the user for error reporting purposes). All lines in the TFS response should end by the <CR><LF> sequence following the standard HTTP convention.

3.3 Error Codes

The possible error codes in the case of plain text output are:

Code	Meaning
0	OK
1	Internal server error.
2	Bad or missing operation.
3	Bad or missing parameters for a given operation.
4	Unknown font (for the <code>fontinfo</code> query).
5	Unknown file (for the <code>fileinfo</code> query).

3.4 Available Queries

Let's look now at the available queries and their responses. As mentioned above, each query is specified by a different value of the "op" key; furthermore, each query may take a set of different parameters in the "key=value" form. This version of the TFS interface specifies the following queries:

tfsversion

Syntax:

`op=tfsversion`

Description:

This query returns the version of the TFS interface supported by the server. Any TFS server which claims conformance a given TFS interface should support all queries specified in that version of the interface, as well as all previous versions of the TFS interface. The interface version is returned as a simple positive integer on a line by its own.

Sample query:

`http://lyra/tfs?op=tfsversion`

Sample response:

0: OK
1

Error codes:

0, 1

fontlist

Syntax:

```
op=fontlist
```

Description:

This query returns a list of fonts known by the server. It is mostly intended to present a nice interface to the TFS to interactive users.

Sample query:

```
http://lyra/tfs?op=fontlist
```

Sample response:

```
0: OK
cmbx10
cmr10
...
cmss10
```

Error codes:

```
0, 1
```

fontinfo

Syntax:

```
op=fontinfo&name=xyz
```

Description:

This query returns information about the font specified by the name parameter. This should be the most frequently called query by client software (eg., DVI drivers) as it enables the client to download and install the font automatically.

The query returns a list of available formats for the requested font. Each format is contained in a single “section” which begins the format name enclosed in square brackets ([...]). Valid formats are: VF (Virtual font format), MF (Metafont format), T1 (Adobe Type 1 format), TT (TrueType format). If a font is virtual, then the TFS response should just include a VF section. If the font is non-virtual, then the TFS response will consist of one or more of the MF, T1 or TT sections. In all cases, each section contains the TFM file as well as *all* files needed to use the font. In the MF section for example, you will see the TFM name, followed by the MF source of the font, followed by any other MF files referenced by the font’s MF source. In the T1 section, you’ll also see the TFM name, followed by the MAP file, followed by any other files referenced in the MAP file. In the TT section, you’ll see the TFM name, followed by the TrueType map file (TTM), followed by any other files referenced by the TTM file. There is obviously some redundancy in the TFS response as the TFM file reference is present in all sections, but this simplifies the task of the TFS client; the general idea is that the client submits a `fontinfo` query and then selects the most convenient format (or the format selected by the user).

Each line within each section pertains to a font file. The format of the line is:

supplier/typeface/fonfile=URL

The `supplier` and `typeface` fields inform the client about the TDS-compliant location of the specified file. The `fontfile` field provides the name of the file, and the URL fields specifies an `http:` or `ftp:` URL for the file; note that the URL cannot be relative, since it may point to any Internet address. It may appear that the `supplier`, `typeface` and filename information is duplicated (since the same information is present in the URL of the example below), but this will not be necessarily true. The specified URL may take any form convenient to the server; the client should *not* try to extract any information about the font file from the URL field. All the pertinent information lies to the left of the equals sign.

An obvious optimization for a TFS client is to download only the files actually missing from the client machine. At the same time, such a client would be advised to download *all* missing files listed in the chosen section. One of the services provided by a TFS is the generation of an explicit dependency list; this information can be hard to acquire (especially for MF files), so the clients should use this service; it will actually save them time and effort.

Sample query:

```
http://lyra/tfs?op=fontinfo&name=cmr10
```

Sample response:

```
0: OK
[MF]
public/cm/cmr10.tfm=http://lyra/tfs/tfm/public/cm/cmr10.tfm
public/cm/cmr10.mf=http://lyra/tfs/mf/public/cm/cmr10.mf
public/cm/cmbase.mf=http://lyra/tfs/mf/public/cm/cmbase.mf
public/cm/accent.mf=http://lyra/tfs/mf/public/cm/accent.mf
public/cm/comlig.mf=http://lyra/tfs/mf/public/cm/comlig.mf
public/cm/greeku.mf=http://lyra/tfs/mf/public/cm/greeku.mf
public/cm/punct.mf=http://lyra/tfs/mf/public/cm/punct.mf
public/cm/roman.mf=http://lyra/tfs/mf/public/cm/roman.mf
public/cm/romand.mf=http://lyra/tfs/mf/public/cm/romand.mf
public/cm/romanl.mf=http://lyra/tfs/mf/public/cm/romanl.mf
public/cm/romanp.mf=http://lyra/tfs/mf/public/cm/romanp.mf
public/cm/romanu.mf=http://lyra/tfs/mf/public/cm/romanu.mf
public/cm/romlig.mf=http://lyra/tfs/mf/public/cm/romlig.mf
public/cm/romspl.mf=http://lyra/tfs/mf/public/cm/romspl.mf
public/cm/romspu.mf=http://lyra/tfs/mf/public/cm/romspu.mf
[T1]
public/cm/cmr10.tfm=http://lyra/tfs/tfm/public/cm/cmr10.tfm
bluesky/cm/cmr10.map=http://lyra/tfs/map/bluesky/cm/cmr10.map
bluesky/cm/cmr10.pfb=http://lyra/tfs/t1/bluesky/cm/cmr10.pfb
bluesky/cm/cmr10.afm=http://lyra/tfs/afm/bluesky/cm/cmr10.afm
[TT]
public/cm/cmr10.tfm=http://lyra/tfs/tfm/public/cm/cmr10.tfm
```

```
bakoma/cm/cmr10.ttm=http://lyra/tfs/ttm/bakoma/cm/cmr10.ttm
bakoma/cm/cmr10.ttf=http://lyra/tfs/ttf/bakoma/cm/cmr10.ttf
bakoma/cm/bkmcm.tte=http://lyra/tfs/tte/bakoma/cm/bkmcm.tte
```

Error codes:

0, 1, 3, 4

fileinfo

Syntax:

op=fileinfo&name=xyz

Description:

This query returns information about the file specified by the name parameter. This could be used by simpler TFS clients (eg., \TeX trying to locate a TFM file, or a MakeTeX . . . script trying to locate some file). The format of the response is analogous to the fontinfo query; it consists of a single line containing information about the specified file.

Sample query:

http://lyra/tfs?op=fileinfo&name=cmr10.tfm

Sample response:

0: OK
public/cm/cmr10.tfm=http://lyra/tfs/tfm/public/cm/cmr10.tfm

Error codes:

0, 1, 3, 5

4 Client requirements

To be written . . .

5 Sample Server Implementation

To be written . . .

6 Sample Client Implementation

To be written . . .

7 Future Plans

Several items will be considered for later versions of the TFS interface; they were left out of the first version of the interface purely for practical reasons.

- **More convenience features in the sample server (eg., forwarding, caching, mirroring, etc).**

The implementation of the extra features in the sample server requires more time and effort than is currently available. We'll try to add these features as soon as possible. It's worth noting that these features can be implemented in the server without affecting at all the interface to the clients.

- **Serving PK files.**

This presupposes the capability to run Metafont, ps2pk, gsftopk or another program on the server machine and it requires a write permission for that program; as such it may present a security risk (denial of service at least) as well as an increased computational burden on the server machine. Therefore, we felt that this feature should not be required by the TFS interface. Nothing however prevents us from experimenting with it as an optional feature in an already existing TFS server: if it becomes widely implemented and deployed, then we can add it to a subsequent version of the TFS interface.

- **Serving other T_EX-related files (eg., T_EX input files).**

This feature would enable even T_EX itself to operate as a TFS client and download any missing input files automatically. It is also a quite simple feature from a mechanical point of view, but we have to deal with the problem that T_EX input files are not required to have globally unique filenames (unlike Metafont input files) in a TDS universe. In general, one would need some sort of context in order to completely and unambiguously specify the requested file. Furthermore, one would need to handle different versions of such files, etc. We will be dealing with such issues in following versions of the TFS interface.

Future versions of the TFS interface are guaranteed to be *totally* compatible with previous versions. A TFS client can be written to conform to a particular version of the TFS interface and that client is assured that it will get appropriate responses from any TFS server that conforms to that particular version of the TFS interface, as well as any later versions. Obviously, any TFS client conforming to the first version of the TFS interface, should be able to converse with any TFS server. If anybody writes a server that claims to support a particular version of the TFS interface, then the server should be able to handle *all* features of that particular version of the TFS interface, as well as all features of *all* previous versions; otherwise, that server *cannot* be called a "TFS server".

8 Conclusion

As described earlier, the installation difficulties of DVI drivers and the portability problems of DVI documents appear to be major impediments to the widespread usage of the

DVI format. Technical authors often convert their DVI documents to other formats (eg., PDF) which are more universally accessible. At the same time, these other formats may be suboptimal for some classes of documents (eg., the PS or PDF formats do not yield good results with fonts produced by Metafont). The TFS project has the potential to completely eliminate these problems with the DVI format: *any* DVI file can be viewed by *any* TFS-aware DVI driver without the need for *any* intervention from the user. This feature should be of great help to authors, publishers, administrators and technical support personnel. Moreover, it avoids duplicating the effort required to install T_EX fonts on multiple computers. Once a TFS administrator sets up the appropriate entries in the database of the TFS server, then the installation will be automatic to all the TFS clients; this is a tremendous time saver particularly in large T_EX installations.

Another not so obvious benefit of the TFS interface is that it provides more freedom to authors of DVI documents. If for example an author wants to use a rarely used font in his document, then he can upload that font to a TFS server (or set up his own TFS server: it's quite simple) and insert a special in the DVI file instructing the DVI driver to use that TFS. The author can then be reasonably certain that his document can be viewed and printed by anybody using a TFS-aware DVI driver.

A [sample TFS server](#) is already in operation, although its bandwidth capabilities are rather limited. One can test this server by using the latest version of dviwin (3.2) available from [CTAN](#) or the [dviwin home page](#).

The potential of the TFS will be fully realized as more DVI drivers become TFS-aware and as more TFS servers around the globe are installed. We are providing free sample implementations both of a TFS server and a TFS client. They can be downloaded from the [TFS project home page](#). We will also try to help with any questions or problems about setting up such servers and clients.