**文件系统中间件规格书**

**File System Middle module(FSMID) spec**

**需求方：**

上海致达智能科技股份有限公司

**执行方：**

朱凌青 (zhulq32@163.com)

**需求内容概述：**

《配电自动化系统应用DL/T634.5101-2002实施细则》规范中的7.6.3章节下的文件传输部分。

设备的主要功能是采集数据，存入存储器，转换数据格式，上传到上位设备。

同时设备还兼有从上位设备下载配置文件和程序文件，存入存储器，以供设备使用。

因此可以将整个系统抽象成三个模块。数据存储模块，文件系统模块，通信模块。本文主要描述的就是文件系统的功能及实现方案

**数据存储模块：**存储设备采集数据，删除旧有的采集数据，存储配置文件和程序文件。

**通信模块：**将已经格式化好，并排列好的采集数据上传给上位设备，将配置文件及程序文件下载到设备。

**文件系统模块：**将数据存储模块内的原始数据按特定算法格式化成字符串，再将多条格式化之后的字符串按特定顺序排列后，以文件形式存放于内存，以备通信模块读取和查询。还可以将通信模块收到的配置文件或程序文件数据组包并存放在内存里，以备数据存储模块读取。

**注意点：**

1. 数据记录是存在存储器中。是二进制，未格式化的。数据并非连续摆放，而是散列的。
2. 由数据记录生成的字符串文件可能长达几十KB。不能同时将多个文件完全加载到内存中。甚至如果能部分加载文件到内存为最佳。
3. 每一条数据记录由于其存储地址不重复，因此可以生成一个唯一的ID以便读取，查询，格式化。
4. 由于数据存储模块不停会有新数据生成。而旧的过时数据会不停被消除。因此文件映射的数据记录数量实时在变化，导致文件的长度和内容实时变化。
5. 通信模块和数据存储模块可能存在同时操作文件的可能性，因此文件系统模块要做好防冲突的保护机制。
6. 部分文件需要将多种数据类型的记录打包在一起。因此并非简单链表一样将新的数据链接。但是文件内的数据类型是有限的，固定的，可以预知的。
7. 当多种数据类型的记录并存于一个文件时，删除数据不一定从头部删除。
8. 目前每一种类型的数据格式化成字符串后的长度是一定的。如果能设计针对每条数据都可以有不定长度的，以应对之后规范的修改则最佳。

**功能实现方案：**

**第一部分：与通信模块交互**

1. 有限个数文件的查询（7.6.3.1-a/b）。查询条件可选（所有文件或特定日期范围）。查询结果包括文件数量，文件名，文件属性(0)，文件大小，文件时间。
2. 特定文件读取操作激活（7.6.3.2-a/b）。激活参数为文件名。返回激活结果（成功与否）。
3. 已激活文件的读取（7.6.3.2-c/d）。读取参数为特定偏移量，特定长度。返回给定参数对应的文件内容。
4. 特定文件写入操作激活（7.6.3.3-a/b）。激活参数为文件名。返回激活结果（成功与否）。
5. 已激活文件的写入（7.6.3.3-c/d）。读取参数为特定偏移量，特定长度，特定数据内容。返回操作结果（成功与否）。

**第二部分：与数据存储模块交互**

1. 当通讯模块需要读取文件时。调用读数据接口（此协议制定），从数据存储模块中读取格式化好的数据记录，并根据上层读取的文件偏移量和长度进行封装。
2. 当通讯模块需要写入文件时。调用写数据接口（此协议制定），向数据存储模块写入内存缓冲中的数据。
3. 当通讯出现错误时。能够调整文件当前读取指针偏移位置，以便错误重传。
4. 当数据存储模块的内容发生改变时。调用钩子函数（此协议制定）。从而修正文件的长度及日期。

**第三部分：内部功能**

1. 以链表形式维护当前系统内的文件列表。以便于目录查询功能。
2. 绑定文件与数据存储模块的关系。
3. 一旦数据存储模块内容发生改变（增/减），触发文件特性的改变（长度/日期）。
4. 读写保护。防止冲突。

**接口概述：**

**第一部分：文件系统模块对象类型**

FSMID\_FHANDLE

文件句柄：被认作唯一的文件UniqueID。需要以此句柄来对文件内容进行操作。

FSMID\_LOG\_HANDLE

数据句柄：被认作一条数据的UniqueID。需要以此句柄来获取该条数据的格式化后的字符串，字符串长度，生成日期。

**第二部分：面向通信模块接口**

FSMID\_Open()

打开/创建一个文件。获取文件句柄。

FSMID\_Close()

关闭一个文件。释放文件句柄及操作权限。

FSMID\_Remove()

删除一个文件。释放文件所占用资源。

FSMID\_Read()

读取一个文件。

FSMID\_Write()

写入一个文件。

FSMID\_Seek()

调整文件读写指针位置。以便错误重传。

FSMID\_Stat()

获取文件状态信息结构体对象。

FSMID\_List()

列出指定目录下的文件及基本信息。

**第三部分：面向数据模块接口**

FSMID\_Open()

打开/创建一个文件。获取文件句柄。

FSMID\_Close()

关闭一个文件。释放文件句柄及操作权限。

FSMID\_Remove()

删除一个文件。释放文件所占用资源。

FSMID\_Register()

注册文件log接口。配置文件有多少组log资源。

FSMID\_Push()

将log句柄压入文件。等同于写入文件。增加文件长度。当log生成时调用。

FSMID\_Pop()

将log句柄推出文件。等同于写入文件。减少文件长度。当log删除时调用。

**第四部分：需要数据模块提供的钩子函数**

fsmid\_get\_data()

特定数据记录生成的字符串。不将格式化字符串功能放在文件系统模块内。

fsmid\_get\_time()

特定数据记录生成的日期时间。用于映射到文件修改日期时间

fsmid\_get\_size()

特定数据记录生成字符串的长度。用于映射到文件大小。

fsmid\_change()

当一个文件由通信模块生成，写入，关闭后。触发数据记录模块去应对此项改变。

**系统工作流程：**

**第一部分：文件系统初始化流程**

文件系统只驻留于内存，因此系统重启时文件系统及其所有文件都需要每次重新建立。

1. 调用FSMID\_Init()初始化文件系统。
2. 查询数据模块中的内容。根据需要调用FSMID\_Open()生成新的文件。
3. 调用FSMID\_Register()将数据模块中的各种类型的数据记录处理钩子函数植入文件。
4. 将数据模块中的记录生成出FSMID\_LOG\_HANDLE句柄。
5. 调用FSMID\_Push()将数据记录句柄推入文件。
6. 更新文件的内容，长度和日期。
7. 循环步骤4~6
8. 调用FSMID\_Close()关闭文件。
9. 调用步骤2~8生成所有记录对应的文件。

**第二部分：新建数据记录流程（LOG->FILE）**

当数据模块生成一条新记录时，如何将其注入到文件中。

1. 数据成一个FSMID\_LOG\_HANDLE 对象。对应该条数据块。
2. 调用FSMID\_Open()打开对应的文件
3. 调用FSMID\_Push()将数据记录句柄推入文件。
4. 更新文件的内容，长度和日期。
5. 调用FSMID\_Close()关闭文件。

**第三部分：删除数据记录流程(LOG->FILE)**

当数据模块删除一条已注入文件的记录时，如何同时删除文件中的记录内容。

1. 数据成一个FSMID\_LOG\_HANDLE 对象。对应该条数据块。
2. 调用FSMID\_Open()打开对应的文件。
3. 调用FSMID\_Pop()将数据记录句柄导出文件。
4. 更新文件的内容，长度和日期。
5. 调用FSMID\_Close()关闭文件。

**第四部分：读取文件内容流程（FILE->COMM）**

当通信模块需要从文件中读取数据记录被格式化后的字符串时。

1. 调用FSMID\_Open()打开对应的文件
2. 如果有需要，调用FSMID\_Seek()调整文件当前指针偏移位置。
3. 调用FSMID\_Read()将数据格式化的数据字符串导出文件。
4. 如有需要，循环步骤2~3。
5. 调用FSMID\_Close()关闭文件。

**第五部分：写入文件内容流程(COMM->FILE)**

当通信模块需要从文件往文件中写入数据时。

1. 调用FSMID\_Open()打开对应的文件。
2. 如果有需要，调用FSMID\_Seek()调整文件当前指针偏移位置。
3. 调用FSMID\_Write()将数据写入文件。
4. 如有需要，循环步骤2~3。
5. 调用FSMID\_Close()关闭文件。

**第六部分：读取文件内容流程(FILE->LOG)**

当数据记录模块需要从文件中读取由通信层写入的文件时。

1. 调用FSMID\_Open()打开对应的文件。
2. 如果有需要，调用FSMID\_Seek()调整文件当前指针偏移位置。
3. 调用FSMID\_Write()将数据写入文件。
4. 如有需要，循环步骤2~3。
5. 调用FSMID\_Close()关闭文件。

**文件系统特性及使用限制：**

1. 由于写入方式完全不同。因此文件被分为两种类型。一种是只能由数据存储模块写入。另一种是只能由通信模块写入。一个文件只能从这两种类型中选择一种来创建。
2. 读取文件不受限制。
3. 当文件被创建为数据存储模块写入时。必须在写入前先初始化。配置文件由多少种数据记录类型拼接。如果存在文件头。则文件头也作为一种数据记录类型在初始化时进行配置。此类文件占用内存极小。即使被读取时，所占内存也仅为格式化后最长记录的长度。
4. 当文件被创建为通信模块写入时。占用内存较大，目前应为整个文件长度。直到被删除。但是文件被通信模块写入并关闭时。会调用钩子函数让数据存储模块处理。数据存储模块处理完之后可以立即删除此文件以释放内存。
5. 此文件系统因为是通过钩子函数来对原始数据进行格式化。因此改换钩子函数可以直接生成出MSG/XML/…等任意格式化的文件。
6. 此文件系统设计简单，因此只能支持最大一级目录检索功能。同时不支持空目录生成。目录仅可作为检索条件的一部分，或者视为文件名的一部分。
7. 此文件系统完全驻留在内存，因此每次开机都需要重新创建所有文件。