ErrorsAndExceptions

This page is an in-progress attempt to document best practice for Parser authors on how to handle problematic files

Allowed Responses

The Parser contract is that a Tika parser will populate the Metadata object, send XML events to the ContentHandler, or throw one of:

- IOException - if the document stream could not be read
- SAXException - if the SAX events could not be processed
- TikaException - if the document could not be parsed

Suggested Responses

Corrupt File

If the file is corrupted in some way, and cannot be processed, a CorruptedFileException (subclass of TikaException) should be thrown (see Parser contract)

File cannot be read

If an IO problem occurs when reading the document, an IOException should be thrown (see Parser contract)

The only exception is if an underlying library throws an IOException to indicate a broken / corrupted file, in which case catching the IOException and re-throwing a CorruptedFileException *may* be done by the parser to give a more helpful error.

"Empty" File (No Text)

If there is no text in the file, either because it's empty (eg 0 byte text file), or because it's a format that doesn't have text (eg an image), then ???

TBC - should the body be opened then immediately closed, or something else?

File is password protected

EncryptedDocumentException (a subtype of TikaException) should be thrown if the file is password protected and no/incorrect password is given.

(A PasswordProvider should be placed on the ParseContext if known, to allow the parsing to proceed without error)

File would require too much memory to process

Where the file format permits predicting the amount of memory required to process it (eg a length field near the start of a block that *must* be fully processed to proceed), and the parser knows this is more than available / permitted, a TikaMemoryLimitException should be thrown to abort the processing.

If it isn't possible to detect this in advance, parsers *should try* to abort with a TikaMemoryLimitException when they spot things going wrong.

Otherwise, there's a reason we suggest using a separate JVM via ForkParser or TikaServer or similar....

Parser can't handle File

If the file is in a sub-format that the parser can't handle (eg parser supports v2 and v3, document is v1, all share the same mimetype), or uses some options that means that parser can't sensibly handle it, then a UnsupportedFormatException should be thrown

Where possible though, the mimetype and/or detection should be updated to prevent an unsupported version of the file format ending up with the parser!

Document Structure is Broken

If something is very broken with the file / file structure, and it will be impossible to output valid XML for it for some reason, then probably a SAXException is the right thing

Document requests no extraction

Some file formats allow extraction, but request politely that it not be done. Depending on your Tika use case (eg forensics vs user facing search), you may wish to abort processing, or you may wish to process + pass the permission details up the chain.

To pass but inform the caller of the access restrictions, populate the https://tika.apache.org/1.19/api/org/apache/tika/metadata/AccessPermissions.html AccessPermissions metadata properties

To abort, eg if the caller requested strict permissions enforcements, throw a AccessPermissionException

Tika-only Responses

These Exceptions should only be thrown by the core of Tika itself, not the underlying parsers

Broken Tika Config

If the user specifies some configuration for Tika, either explicitly or implicitly (eg extension resources with the right magic specific name on the classpath), and the config supplied is invalid and can't be worked around, a TikaConfigException will be thrown during the initialisation of Tika

Zero byte length file

Tika can provide a suggested mime type for these, but not extract any content. Caused typically by *Right Click -> New* or *touch*, or an error in getting the file to where it is now.

If passed one of these for detection, no error should be given, and a filename-only answer supplied

If passed one of these for parsing, a ZeroByteFileException should be thrown