POIFS Use Cases

by Marc Johnson

1. POIFS Use Cases

1.1. Use Case 1: Read existing file system

Primary Actor:	POIFS client
Scope:	POIFS
Level:	Summary
Stakeholders and Interests:	POIFS client- wants to read content of file system POIFS - understands POIFS file system
Precondition:	None
Minimal Guarantee:	None
Main Success Guarantee:	1. POIFS client requests POIFS to read a POIFS file system, providing an `InputStream` containing POIFS file system in question. 2. POIFS reads from the `InputStream` in 512 byte blocks. 3. POIFS verifies that the first block begins with the well known signature (`0xE11AB1A1E011CFD0`) 4. POIFS reads the Block Allocation Table from the first block and, if necessary, from the XBAT blocks. 5. POIFS obtains the start block of the Property Table and reads the Property Table (use case 9, read file) 6. POIFS reads the individual entries in the Property Table 7. POIFS obtains the start block of the Small Block Allocation Table and reads the Small Block Allocation Table (use case 9, read file) 8. POIFS obtains the start block of the Small

	Block store from the first entry in the Property Table and reads the Small Block Array (use case 9, read file)
Extensions:	2a. If the last block read is not a 512 byte block, the `InputStream` is not that of a POIFS file system, and POIFS throws an appropriate exception. 3a. If the signature is incorrect, the `InputStream` is not that of a POIFS file system, and POIFS throws an appropriate exception.

1.2. Use Case 2: Write file system

Primary Actor:	POIFS client
Scope:	POIFS
Level:	Summary
Stakeholders and Interests:	POIFS client- wants to write file system out. POIFS - knows how to write file system out.
Precondition:	File system has been read (use case 1, read existing file system) and subsequently modified (use case 4, replace file in file system; use case 5, delete file from file system; or use case 6, write new file to file system; in any combination) or File system has been created (use case 3, create new file system)
Minimal Guarantee:	None
Main Success Guarantee:	1. POIFS client provides an `OutputStream` to write the file system to. 2. POIFS gets the sizes of the Property Table and each file in the file system. 3. If any files in the file system requires storage in a Small Block Array, POIFS creates a Small Block Array of sufficient size to hold all of the small files. 4. POIFS calculates the number of big blocks needed to hold all of the large files, the Property Table, and, if necessary, the Small Block Array and the Small Block Allocation Table. 5. POIFS creates a set of big blocks sufficient to

	store the Block Allocation Table 6. POIFS creates and writes the header block 7. POIFS writes out the XBAT blocks, if needed. 8. POIFS writes out the Small Block Array, if needed 9. POIFS writes out the Small Block Allocation Table, if needed 10. POIFS writes out the Property Table 11. POIFS writes out the large files, if needed 12. POIFS closes the `OutputStream`.
Extensions:	6a. Exceptions writing to the `OutputStream` will be propagated back to the POIFS client. 7a. Exceptions writing to the `OutputStream` will be propagated back to the POIFS client. 8a. Exceptions writing to the `OutputStream` will be propagated back to the POIFS client. 9a. Exceptions writing to the `OutputStream` will be propagated back to the POIFS client. 10a. Exceptions writing to the `OutputStream` will be propagated back to the POIFS client. 11a. Exceptions writing to the `OutputStream` will be propagated back to the POIFS client. 12a. Exceptions closing the `OutputStream` will be propagated back to the POIFS client.

1.3. Use Case 3: Create new file system

Primary Actor:	POIFS client
Scope:	POIFS
Level:	Summary
Stakeholders and Interests:	POIFS client- wants to create a new file system POIFS - knows how to create a new file system
Precondition:	None
Minimal Guarantee:	None
Main Success Guarantee:	POIFS creates an empty Property Table.
Extensions:	None

1.4. Use Case 4: Replace file in file system

Primary Actor:	POIFS client

Scope:	POIFS
Level:	Summary
Stakeholders and Interests:	1. POIFS client- wants to replace an existing file in the file system 2. POIFS - knows how to manage the file system
Precondition:	Either The file system has been read (use case 1, read existing file system) and a file has been extracted from the file system (use case 7, read existing file from file system) or The file system has been created (use case 3, create new file system) and a file has been written to the file system (use case 6, write new file to file system)
Minimal Guarantee:	None
Main Success Guarantee:	1. POIFS discards storage of the existing file. 2. POIFS updates the existing file's entry in the Property Table 3. POIFS stores the new file's data
Extensions:	1a. POIFS throws an exception if the file does not exist.

1.5. Use Case 5: Delete file from file system

Primary Actor:	POIFS client
Scope:	POIFS
Level:	Summary
Stakeholders and Interests:	* POIFS client- wants to remove a file from a file system * POIFS - knows how to manage the file system
Precondition:	Either The file system has been read (use case 1, read existing file system) and a file has been extracted from the file system (use case 7, read existing file from file system) or The file system has been created (use case 3,

	create new file system) and a file has been written to the file system (use case 6, write new file to file system)
Minimal Guarantee:	None
Main Success Guarantee:	1. POIFS discards the specified file's storage. 2. POIFS discards the file's Property Table entry.
Extensions:	1a. POIFS throws an exception if the file does not exist.

1.6. Use Case 6: Write new file to file system

Primary Actor:	POIFS client
Scope:	POIFS
Level:	Summary
Stakeholders and Interests:	* POIFS client- wants to add a new file to the file system * POIFS - knows how to manage the file system
Precondition:	The specified file does not yet exist in the file system
Minimal Guarantee:	None
Main Success Guarantee:	1. The POIFS client provides a file name 2. POIFS creates a new Property Table entry for the new file 3. POIFS provides the POIFS client with an `OutputStream` to write to. 4. The POIFS client writes data to the provided `OutputStream`. 5. The POIFS client closes the provided `OutputStream` 6. POIFS updates the Property Table entry with the new file's size
Extensions:	1a. POIFS throws an exception if a file with the specified name already exists in the file system. 1b. POIFS throws an exception if the file name is too long. The limit on file name length is 31 characters.

1.7. Use Case 7: Read existing file from file system

Primary Actor:	POIFS client
Scope:	POIFS
Level:	Summary
Stakeholders and Interests:	* POIFS client- wants to read a file from the file system * POIFS - knows how to manage the file system
Precondition:	* The file system has been read (use case 1, read existing file system) or has been created and written to (use case 3, create new file system; use case 6, write new file to file system). * The specified file exists in the file system.
Minimal Guarantee:	None
Main Success Guarantee:	* The POIFS client provides the name of a file to be read * POIFS provides an `InputStream` to read from. * The POIFS client reads from the `InputStream`. * The POIFS client closes the `InputStream`.
Extensions:	1a. POIFS throws an exception if no file with the specified name exists.

1.8. Use Case 8: Read file system directory

Primary Actor:	POIFS client
Scope:	POIFS
Level:	Summary
Stakeholders and Interests:	* POIFS client- wants to know what files exist in the file system * POIFS - knows how to manage the file system
Precondition:	The file system has been read (use case 1, read existing file system) or created (use case 3, create new file system)

Minimal Guarantee:	None
Main Success Guarantee:	1. The POIFS client requests the file system directory. 2. POIFS returns an `Iterator`. The `Iterator` will not include the root entry in the Property Table, and may be an `Iterator` over an empty `Collection`.
Extensions:	None

1.9. Use Case 9: Read file

Primary Actor:	POIFS
Scope:	POIFS
Level:	Summary
Stakeholders and Interests:	POIFS - POIFS needs to read a file, or something resembling a file (i.e., the Property Table, the Small Block Array, or the Small Block Allocation Table)
Precondition:	None
Minimal Guarantee:	None
Main Success Guarantee:	1. POIFS begins with a start block, a file size, and a flag indicating whether to use the Big Block Allocation Table or the Small Block Allocation Table 2. POIFS returns an `InputStream`. 3. Reads from the `InputStream` are performed by walking the specified Block Allocation Table and reading the blocks indicated. 4. POIFS closes the `InputStream` when finished reading the file, or its client wants to close the `InputStream`.
Extensions:	3a. An exception will be thrown if the specified Block Allocation Table is corrupt, as evidenced by an index pointing to a non-existent block, or by a chain extending past the known size of the file.

1.10. Use Case 10: Rename existing file in the file system

Primary Actor:	POIFS client

Scope:	POIFS
Level:	Summary
Stakeholders and Interests:	* POIFS client- wants to rename an existing file in the file system. * POIFS - knows how to manage the file system.
Precondition:	* The file system is has been read (use case 1, read existing file system) or has been created and written to (use case 3, create new file system; use case 6, write new file to file system. * The specified file exists in the file system. * The new name for the file does not duplicate another file in the file system.
Minimal Guarantee:	None
Main Success Guarantee:	1. POIFS updates the Property Table entry for the specified file with its new name.
Extensions:	* 1a. If the old file name is not in the file system, POIFS throws an exception. * 1b. If the new file name already exists in the file system, POIFS throws an exception. * 1c. If the new file name is too long (the limit is 31 characters), POIFS throws an exception.