Apache NiFi 3

Managing a Data Flow

Date of Publish: 2020-12-15



https://docs.cloudera.com/

Contents

Command and Control of the DataFlow	3
Starting a Component	3
Stopping a Component	3
Enabling/Disabling a Component	4
Remote Process Group Transmission	4
Individual Port Transmission	4

Encrypted Content Repository	6
What is it?	7
How does it work?	7
StaticKeyProvider	7
FileBasedKeyProvider	7
Key Rotation	8
Writing and Reading Content Claims	8
Potential Issues	8

Encrypted FlowFile Repository	9
What is it?	
How does it work?	
StaticKeyProvider	
FileBasedKeyProvider	
Key Rotation	
Writing and Reading FlowFiles	
Potential Issues	
Experimental Warning	12

Other	Ianagement Features12

Command and Control of the DataFlow

When a component is added to the NiFi canvas, it is in the Stopped state. In order to cause the component to be triggered, the component must be started. Once started, the component can be stopped at any time. From a Stopped state, the component can be configured, started, or disabled.

Starting a Component

In order to start a component, the following conditions must be met:

- The component's configuration must be valid.
- All defined Relationships for the component must be connected to another component or auto-terminated.
- The component must be stopped.
- The component must be enabled.
- The component must have no active tasks. For more information about active tasks, see the "Anatomy of ..." sections under Monitoring of DataFlow.

Components can be started by selecting all of the components to start and then clicking the "Start" button (



) in the Operate Palette or by right-clicking a single component and choosing Start from the context menu.

If starting a Process Group, all components within that Process Group (including child Process Groups) will be started, with the exception of those components that are invalid or disabled.

Once started, the status indicator of a Processor will change to a Play symbol (

		٢		
L	r			
ļ	r			
ļ	r			
ļ	ľ			

).

Stopping a Component

A component can be stopped any time that it is running. A component is stopped by right-clicking on the component and clicking Stop from the context menu, or by selecting the component and clicking the "Stop" button (



) in the Operate Palette.

If a Process Group is stopped, all of the components within the Process Group (including child Process Groups) will be stopped.

Once stopped, the status indicator of a component will change to the Stop symbol (

).

Stopping a component does not interrupt its currently running tasks. Rather, it stops scheduling new tasks to be performed. The number of active tasks is shown in the top-right corner of the Processor (See Anatomy of a Processor for more information).

Enabling/Disabling a Component

When a component is enabled, it is able to be started. Users may choose to disable components when they are part of a dataflow that is still being assembled, for example. Typically, if a component is not intended to be run, the component is disabled, rather than being left in the Stopped state. This helps to distinguish between components that are intentionally not running and those that may have been stopped temporarily (for instance, to change the component's configuration) and inadvertently were never restarted.

When it is desirable to re-enable a component, it can be enabled by selecting the component and clicking the "Enable" button



) in the Operate Palette. This is available only when the selected component or components are disabled. Alternatively, a component can be enabled by checking the checkbox next to the "Enabled" option in the Settings tab of the Processor configuration dialog or the configuration dialog for a Port.

Once enabled, the component's status indicator will change to either Invalid (



) or Stopped (



), depending on whether or not the component is valid.

A component is then disabled by selecting the component and clicking the "Disable" button (



) in the Operate Palette, or by clearing the checkbox next to the "Enabled" option in the Settings tab of the Processor configuration dialog or the configuration dialog for a Port.

Only Ports and Processors can be enabled and disabled.

Remote Process Group Transmission

Remote Process Groups provide a mechanism for sending data to or retrieving data from a remote instance of NiFi. When a Remote Process Group (RPG) is added to the canvas, it is added with the Transmission Disabled, as indicated by the icon (



) in the top-left corner. When Transmission is Disabled, it can be enabled by right-clicking on the RPG and clicking the "Enable transmission" menu item. This will cause all ports for which there is a Connection to begin transmitting data. This will cause the status indicator to then change to the Transmission Enabled icon (

).

If there are problems communicating with the Remote Process Group, a Warning indicator (

A

) may instead be present in the top-left corner. Hovering over this Warning indicator with the mouse will provide more information about the problem.

Individual Port Transmission

There are times when the DFM may want to either enable or disable transmission for only a specific port within the Remote Process Group. This can be accomplished by right-clicking on the Remote Process Group and choosing the "Manage remote ports" menu item. This provides a configuration dialog from which ports can be configured:

Rem	ote Process	Group Port	S				
Name NiFi Flo	w			URLs http://l	ocalhost:8080/r	iifi, http://localho	st:8081/nifi
Input p	orts			Output	ports		
	Input1 No description spec	cified.			Output1 No description spec	cified.	
	Concurrent Tas 1	ks 😧	Compressed No	ø	Concurrent Tas 1	ks 😧	Compressed Yes
	Batch Settings	0			Batch Settings	0	
	Count 10	Size 10 KB	Duration 10 sec		Count No value set	Size No value set	Duration No value set
	Input2	rified			Output2	cified	
	Concurrent Tas	ks 🛛	Compressed		Concurrent Tasks 🕜		Compressed
	1 Datah Gaminan	•	No		1 Datable Cattion of	•	No
	Count	Size	Duration		Batch Settings Count	Size	Duration
	No value set	No value set	No value set		No value set	No value set	No value set
							CLOSE

The left-hand side lists all of the Input Ports that the remote instance of NiFi allows data to be sent to. The right-hand side lists all of the Output Ports from which this instance is able to pull data. If the remote instance is using secure communications (the URL of the NiFi instance begins with https://, rather than http://), any ports that the remote instance has not made available to this instance will not be shown.

Note: If a port that is expected to be shown is not shown in this dialog, ensure that the instance has proper permissions and that the Remote Process Group's flow is current. This can be checked by closing the Remote Process Group Ports dialog and looking at the bottom-left corner of the Remote Process Group. The date and time when the flow was last refreshed is displayed. If the flow appears to be outdated, it can be updated by right-clicking on the Remote Process Group and selecting "Refresh remote". (See Anatomy of a Remote Process Group for more information).

Each port is shown with its Name, its Description, configured number of Concurrent Tasks, and whether or not data sent to this port will be Compressed. Additionally, the port's configured Batch Settings (Count, Size and Duration) are displayed. To the left of this information is a toggle switch to turn the port on or off. Ports that have no connections attached to them are grayed out:

)

Remo	te Process	Group Port	S				
Name NiFi Flov		urrently Trar	nsmitting	URLs http://l	ocalhost:8080/r	nifi, http://localho	ost:8081/nifi
Input po	rts			Output	ports		
	Input1 No description spe	cified.			Output1 No description spe	cified.	
	Concurrent Tas 1	sks 😧	Compressed No	B	Concurrent Tas 1	ks 🛿	Compressed Yes
	Batch Settings	0		_ <u>/</u>	Batch Settings	0	
	Count	Size	Duration	_	Count	Size	Duration
	10	10 KB	10 sec	Edit	No value set	No value set	No value set
	Input2 No description spe	cified.			Output2 No description spe	cified.	
	Concurrent Tas	sks 🕜	Compressed		Concurrent Tasks 🔞		Compressed
Т	1		No		1		No
	Batch Settings	0			Batch Settings	0	
	Count	Size	Duration		Count	Size	Duration
	No value set	No value set	No value set		No value set	No value set	No value set
Not C	onnected						
							CLOSE

The on/off toggle switch provides a mechanism to enable and disable transmission for each port in the Remote Process Group independently. Those ports that are connected but are not currently transmitting can be configured by clicking the pencil icon

below the on/off toggle switch. Clicking this icon will allow the DFM to change the number of Concurrent Tasks, whether or not compression should be used when transmitting data to or from this port, and Batch Settings.

For an Input Port, the batch settings control how NiFi sends data to the remote input port in a transaction. NiFi will transfer flow files, as they are queued in incoming relationships, until any of the limits (Count, Size, Duration) is met. If none of the settings are configured, a 500 milliseconds batch duration is used by default.

For an Output Port, the batch settings tells the remote NiFi how NiFi prefers to receive data from the remote output port in a transaction. The remote NiFi will use the specified settings (Count, Size, Duration) to control the transfer of flow files. If none of the settings are configured, a 5 seconds batch duration is used by default.

Encrypted Content Repository

Ø

While OS-level access control can offer some security over the flowfile content data written to the disk in a repository, there are scenarios where the data may be sensitive, compliance and regulatory requirements exist, or NiFi is running on hardware not under the direct control of the organization (cloud, etc.). In this case, the content repository allows for all data to be encrypted before being persisted to the disk. For more information on the internal workings of the content repository, see NiFi In-Depth - Content Repository.

What is it?

The EncryptedFileSystemRepository is a new implementation of the content repository which encrypts all content data before it is written to the repository. This allows for storage on systems where OS-level access controls are not sufficient to protect the data while still allowing querying and access to the data through the NiFi UI/API.

How does it work?

The FileSystemRepository was introduced in NiFi 0.2.1 and provided the only persistent content repository implementation. The encrypted version wraps that implementation with functionality to return to the Session (usually StandardProcessSession) a special OutputStream/InputStream which encrypt and decrypt the serialized bytes respectively. This allows all components to continue interacting with the content repository interface in the same way as before and continue operating on content data in a streaming manner, without requiring any changes to handle the data protection.

The fully qualified class org.apache.nifi.content.EncryptedFileSystemRepository is specified as the content repository implementation in nifi.properties as the value of nifi.content.repository.implementation. In addition, new properties must be populated to allow successful initialization.

StaticKeyProvider

The StaticKeyProvider implementation defines keys directly in nifi.properties. Individual keys are provided in hexadecimal encoding. The keys can also be encrypted like any other sensitive property in nifi.properties using the ./ encrypt-config.sh tool in the NiFi Toolkit.

The following configuration section would result in a key provider with two available keys, "Key1" (active) and "AnotherKey".

FileBasedKeyProvider

The FileBasedKeyProvider implementation reads from an encrypted definition file of the format:

```
key1=NGCpDpxBZNN0DBodz0p1SDbTjC2FG5kp1pCmdUKJ1xxtcMSo6GC4fMlTyy1mPeKOxzLut3DRX
+51j6PC05SznA==
key2=GYxPbMMDbnraXs09eGJudAM5jTvVYp05XtImkAg4JY4rIbmHOiVUUI6OeOf7ZW
+hH42jtPgNW9pSkkQ9HWY/vQ==
key3=SFe11xuz7J89Y/IQ7YbJPOL0/YKZRFL/
VUxJgEHxx1Xpd/8ELA7wwN59K1KTr3BURCcFP5YGmwrSKfr4OE4Vlg==
key4=kZprfcTSTH69UuOU3jMkZfrtiVR/eqWmmbdku3bQcUJ/
+UToecNB51zOVEMBChyEXppyXXC35Wa6GEXFK6PMKw==
key5=c6FzfnKm7UR7xqI2NFpZ+fEKBfSU7+1NvRw
+XWQ9U39MONWqk5gvoyOCdFR1kUgeg46jrN5dGXk13sRqE0GETQ==
```

Each line defines a key ID and then the Base64-encoded cipher text of a 16 byte IV and wrapped AES-128, AES-192, or AES-256 key depending on the JCE policies available. The individual keys are wrapped by AES/GCM encryption using the master key defined by nifi.bootstrap.sensitive.key in conf/bootstrap.conf.

Data Protection vs. Key Protection

Even though the flowfile content is encrypted using AES/CTR to handle streaming data, if using the Config Encrypt Tool or FileBasedKeyProvider, those keys will be protected using AES/GCM to provide authenticated encryption over the key material.

Key Rotation

Simply update nifi.properties to reference a new key ID in nifi.content.repository.encryption.key.id. Previouslyencrypted content claims can still be decrypted as long as that key is still available in the key definition file or nifi.content.repository.encryption.key.id.<OldKeyID> as the key ID is serialized alongside the encrypted content.

Writing and Reading Content Claims

Once the repository is initialized, all content claim write operations are serialized using RepositoryObjectStreamEncryptor (the only currently existing implementation is RepositoryObjectAESCTREncryptor) to an OutputStream. The actual implementation is EncryptedContentRepositoryOutputStream, which encrypts the data written by the component via StandardProcessSession inline and the encryption metadata (keyId, algorithm, version, IV) is serialized and prepended. The complete OutputStream is then written to the repository on disk as normal.

							740506000	05 02		
235476	ORORGEER	02362410	38350200	A3E7D201	30162128	39505480	03611832	30134627	RECE3366	0 6\$ 8> 9 1(9P7 a 20 0' 3f
2255470	JENCZCDE	02302410	07054022	25054257	45206174	10104007	05011052	2401000	E0D79447	. 0. 03 8>9 :(9F2. 0 20 0
232217		2001002009	4D960275	CE792476	21009501	40104007	6065CA01	6027D6CD	DC90510D	$0 \rightarrow i_{2}M + x_{2}x_{3} + S_{1} = S_{1}^{2} + O_{1}^{2} = O_{1}^{2}$
235584	004FA4AD	131CD0C3	40009373 6CD1525A	71DDEE1D	53175546		AG180448	695700CD	210005100	
225620	0110E7AE	COVEEDVO		974D96D0	20005000	2002000	A0100440	C462EE71	01654724	W F X (75 h a nC4
235656	011957A5	C3AF3DA3	009275DE	C1D2E5A9	12852013	624154D2	EGAEDEOC		921C6DCE	· · · · · · · · · · · · · · · · · · ·
235602	04381300	72544965	0D1703BC	03702267	42023943	62413403	R763/170			μητάο ν" KSV cAp '
235728	38318502	FDE21285	00110013	F2RR2238	36277CR0	E64D1B3E	24558540	5C4E0E31	AE533CEA	81 "86'L M \\$ \N 1 S-
235764	4D9R8F96	79R4C7C3	79405848	44000418	8DC1E015	184R7315	D3F14R28	70358478	551143E8	M V V X D Ks K(15 XII (
235800	F85R0AQA	1311800R	ASRECRSA	C2051E63	E2803607	83801/10	ED155810	132CEE4E	350EC48E	
235836	8CD3ED6A	65C6C5B3	98046292	A2E1E3R8	QECR2R7A	16560467	D9487087	7FR28AC1	5E5474ED	$ie h \pm 7$ Hn $\Lambda7$
235872	A47102CF	5766R6A2	B6CD26B6	44003044	33341002	9546CEA7	1672B302	4941 1019	B7967D9C	a Wf & $1 - D3$ r TA \clubsuit 3
235908	RE4RE557	5382C6RE	AC81D069	R9639D74	F8R1R398	F0D02842	01909F49	Π7ΔΔE742	5645B475	KWS i c z (B T BV u
235944	BD06FAFC	C1R4RDF8	CDA290D5	E43C67E7	8ARC59RR	4696F4F4	7349D73F	6D9F2DCD	46097406	$< q$ Y E s $\rightarrow m - E z$
235980	8480F439	2C8D5F1B	0085924	CA44C50F	FDRF9RR2	4F9468F5	69264706	F7184F89	BE947107	9. Y*D ki& a
236016	AB0F92B6	CAFE7AAD	C6671F4D	9F21CF49	9ADB64FB	D20036DC	42435107	F43FB274	BBE166EE	a M b T d b B 0 $>$ t f
236052	60700879	5B1F0BE6	DAC65404	FA54CD18	6401898E	74483909	1104A7AD	CF0000C3	306579A7	m}.v[TT. dzH90ev.
236088	13EE834E	9AE9AAEF	3892B434	7F93D1BC	51B751B2	BC3AB1BD	FF5C0066	2E765297	37A0D5B2	
236124	674FC77B	BB780B2D	7F95AD2F	CEAB9375	289B7405	6B783F9D	D5C02187	7D4A562C	646F8679	$aN_{x} = \dots + u(t kx > \dots + 1)V_{x} dn_{x}$
236160	54A495D6	D08533FD	309049EF	41ADEB3A	8CD8A1BA	3E1A35B8	0D20D823	306F3C62	58FA382D	T3.0.I.A:> 5#0o <bx.8-< td=""></bx.8-<>
236196	8A865F91	AC768E30	717FC82C	A336BC9A	ØAD29620	99B82D5E	140ABA2E	268B9916	D3988EØA	v.0a6^&
236232	199534D3	A26DE78A	70A3C766	072E7F72	2AB31CCA	380000AC	ED000573	72003F6F	72672E61	.4mpf. r*8 sr ?org.a
236268	70616368	652E6E69	66692E73	65637572	6974792E	7265706F	7369746F	72792E53	74726561	pache.nifi.security.repository.Strea
236304	6D696E67	456E6372	79707469	6F6E4D65	74616461	74611846	6982894D	442C0200	00787200	mingEncryptionMetadata FiMD, xr
236340	466F7267	2E617061	6368652E	6E696669	2E736563	75726974	792E7265	706F7369	746F7279	Forg.apache.nifi.security.repository
236376	2E526570	6F736974	6F72794F	626A6563	74456E63	72797074	696F6E4D	65746164	6174619F	.RepositoryObjectEncryptionMetadata.
236412	4328584E	DFDF0802	00054900	10636970	68657242	7974654C	656E6774	684C0009	616C676F	C(XN I cipherByteLengthL algo
236448	72697468	6D740012	4C6A6176	612F6C61	6E672F53	7472696E	673B5B00	07697642	79746573	rithmt Ljava/lang/String;[ivBytes
236484	7400025B	424C0005	6B657949	6471007E	00024C00	07766572	73696F6E	71007E00	027870FF	t [BL keyIdg ~ L versiong ~ xp.
236520	FFFFFF74	00114145	532F4354	522F4E6F	50616464	696E6775	7200025B	42ACF317	F8060854	t AES/CTR/NoPaddingur [B T
236556	E0020000	78700000	00104207	611FD7A6	81A8E8EA	FFB8FFA4	9C117400	024B3174	00027631	. xp B a t K1t v1
236592										
UTF-8	٥	(select s	ome data)							-+
						0x18E8	3F out of 0x390	C30 bytes		

On content claim read, the process is reversed. The encryption metadata (RepositoryObjectEncryptionMetadata) is parsed and used to decrypt the serialized bytes, which are then deserialized into a CipherInputStream object. The delegation to the normal repository file system interaction allows for "random-access" (i.e. immediate seek without decryption of unnecessary content claims).

Within the NiFi UI/API, there is no detectable difference between an encrypted and unencrypted content repository. The Provenance Query operations to view content work as expected with no change to the process.

Potential Issues

• Switching between unencrypted and encrypted repositories

- If a user has an existing repository (FileSystemRepository) that is not encrypted and switches their configuration to use an encrypted repository, the application writes an error to the log but starts up. However, previous content claims are not accessible through the provenance query interface and new content claims will overwrite the existing claims. The same behavior occurs if a user switches from an encrypted repository to an unencrypted repository. Automatic roll-over is a future effort (https://issues.apache.org/jira/browse/NIFI-6783) but NiFi is not intended for long-term storage of content claims so the impact should be minimal. There are two scenarios for roll-over:
 - Encrypted # unencrypted if the previous repository implementation was encrypted, these claims should be handled seamlessly as long as the key provider available still has the keys used to encrypt the claims (see *Key Rotation*.)
 - Unencrypted # encrypted if the previous repository implementation was unencrypted, these claims should be handled seamlessly as the previously written claims simply need to be read with a plaintext InputStream and then be written back with the EncryptedContentRepositoryOutputStream
- There is also a future effort to provide a standalone tool in NiFi Toolkit to encrypt/decrypt an existing content repository to make the transition easier. The translation process could take a long time depending on the size of the existing repository, and being able to perform this task outside of application startup would be valuable (https://issues.apache.org/jira/browse/NIFI-6783).
- Multiple repositories No additional effort or testing has been applied to multiple repositories at this time. It is possible/likely issues will occur with repositories on different physical devices. There is no option to provide a heterogenous environment (i.e. one encrypted, one plaintext repository).
- Corruption when a disk is filled or corrupted, there have been reported issues with the repository becoming corrupted and recovery steps are necessary. This is likely to continue to be an issue with the encrypted repository, although still limited in scope to individual claims (i.e. an entire repository file won't be irrecoverable due to the encryption). Some testing has been performed on scenarios where disk space is exhausted. While the flow can no longer write additional content claims to the repository in that case, the NiFi application continues to function properly, and successfully written content claims are still available via the Provenance Query operations. Stopping NiFi and removing the content repository (or moving it to a larger disk) resolves the issue.

Encrypted FlowFile Repository

While OS-level access control can offer some security over the flowfile attribute and content claim data written to the disk in a repository, there are scenarios where the data may be sensitive, compliance and regulatory requirements exist, or NiFi is running on hardware not under the direct control of the organization (cloud, etc.). In this case, the flowfile repository allows for all data to be encrypted before being persisted to the disk. For more information on the internal workings of the flowfile repository, see NiFi In-Depth - FlowFile Repository.

What is it?

The EncryptedSequentialAccessWriteAheadLog is a new implementation of the flowfile write-ahead log which encrypts all flowfile attribute data before it is written to the repository. This allows for storage on systems where OS-level access controls are not sufficient to protect the data while still allowing querying and access to the data through the NiFi UI/API.

How does it work?

The SequentialAccessWriteAheadLog was introduced in NiFi 1.6.0 and provided a faster flowfile repository implementation. The encrypted version wraps that implementation with functionality to transparently encrypt and decrypt the serialized RepositoryRecord objects during file system interaction. During all writes to disk (swapping, snapshotting, journaling, and checkpointing), the flowfile containers are serialized to bytes based on a schema, and this serialized form is encrypted before writing. This allows the snapshot handler to continue interacting with the

flowfile repository interface in the same way as before and continue operating on flowfile data in a random access manner, without requiring any changes to handle the data protection.

The fully qualified class org.apache.nifi.wali.EncryptedSequentialAccessWriteAheadLog is specified as the flowfile repository write-ahead log implementation in nifi.properties as the value of nifi.flowfile.repository.wal.implementation. In addition, new properties must be populated to allow successful initialization.

StaticKeyProvider

The StaticKeyProvider implementation defines keys directly in nifi.properties. Individual keys are provided in hexadecimal encoding. The keys can also be encrypted like any other sensitive property in nifi.properties using the ./ encrypt-config.sh tool in the NiFi Toolkit.

The following configuration section would result in a key provider with two available keys, "Key1" (active) and "AnotherKey".

FileBasedKeyProvider

The FileBasedKeyProvider implementation reads from an encrypted definition file of the format:

```
key1=NGCpDpxBZNN0DBodz0p1SDbTjC2FG5kp1pCmdUKJ1xxtcMSo6GC4fMlTyy1mPeKOxzLut3DRX
+51j6PCO5SznA==
key2=GYxPbMMDbnraXs09eGJudAM5jTvVYp05XtImkAg4JY4rIbmHOiVUUI6OeOf7ZW
+hH42jtPgNW9pSkkQ9HWY/vQ==
key3=SFe11xuz7J89Y/IQ7YbJPOL0/YKZRFL/
VUxJgEHxx1Xpd/8ELA7wwN59K1KTr3BURCcFP5YGmwrSKfr4OE4V1g==
key4=kZprfcTSTH69UuOU3jMkZfrtiVR/eqWmmbdku3bQcUJ/
+UToecNB51zOVEMBChyEXppyXXC35Wa6GEXFK6PMKw==
key5=c6FzfnKm7UR7xq12NFpZ+fEKBfSU7+1NvRw
+XWQ9U39MONWqk5gvoyOCdFR1kUgeg46jrN5dGXk13sRqE0GETQ==
```

Each line defines a key ID and then the Base64-encoded cipher text of a 16 byte IV and wrapped AES-128, AES-192, or AES-256 key depending on the JCE policies available. The individual keys are wrapped by AES/GCM encryption using the master key defined by nifi.bootstrap.sensitive.key in conf/bootstrap.conf.

Key Rotation

Simply update nifi.properties to reference a new key ID in nifi.flowfile.repository.encryption.key.id. Previouslyencrypted flowfile records can still be decrypted as long as that key is still available in the key definition file or nifi.flowfile.repository.encryption.key.id.<OldKeyID> as the key ID is serialized alongside the encrypted record.

Writing and Reading FlowFiles

Once the repository is initialized, all flowfile record write operations are serialized using RepositoryObjectBlockEncryptor (the only currently existing implementation is RepositoryObjectAESGCMEncryptor) to the provided DataOutputStream. The original stream is swapped with a temporary wrapped stream, which encrypts the data written by the wrapped serializer/deserializer via EncryptedSchemaRepositoryRecordSerde inline and the encryption metadata (keyId, algorithm, version, IV, cipherByteLength) is serialized and prepended. The complete length and encrypted bytes are then written to the original DataOutputStream on disk as normal.

11748 0A6995616 FDACCE9A BB1242F5 258EBAA0 20781D86 2A620000 0224ACED 00057372 00416F72 67262170 61636865 11792 22666966 692E7365 63757269 74792E72 6576673 69746F72 7722E62626 6636822 42666F63 692E7365 ionHetadata6 x Forg.apache.nifi.se 11924 40557461 6417461 94432858 4EDF198 020007872 0006369 766726372 7727074 7460004 09616677 0740002 S8424C60 071472 7722567 77972777 7490007 1000777 10007757 1007787 100786787 100786787 7727074 100786787 7727074 100786787 100786787 100786787 100786787 100786787 100786787 100786787 100786787 100786787 100786787 100786787 100786787 100786787 100786787 1007868 1007877 10078688 1017975 100786787 100786787 100786787 100786787 100786787 1178977 11886 11787777 10078689789 10078778 10078787 10078778 100787787 1										iournal			
 11748 0A695G16 FDACCE9A BB1242F5 258EBAA0 207B1086 2A620000 0224ACED 0005732 00416F72 672E6170 61636865 11792 ZEEG606 60227365 6375250 74792E72 65706F73 69746F72 972E526 F6638E2 426C6F63 68456E3 72797070 11836 696F6E4D 65746164 61746136 C69C49D5 97A81F02 00007872 00466F72 72725 4250C6 6374566 63727970 11846 63757269 74792E72 25706773 69746F72 072E5265 706F736 9746F7279 4F626A65 5374456E 63727970 11924 40557461 64617461 94732585 4EDDF08 020009849 001063079 0706557 2739456 46536279 74881C00 00600001 74001141 45532F47 74840C0 00600001 74001141 45532F47 43402F4E 12012 05686579 49647100 7E000274 00077665 7273696 6E71007E 00027870 00000001 74001141 45532F47 43402F4E 12012 05686579 49647100 7E000274 00077665 7273696 6E71007E 00027870 00000001 74001141 45532F47 43402F4E 12106 0568567 4592 4542CF3 171880608 54E00200 00787080 0000010 74001141 5532F47 43402F4E 12144 66A62E48 701107F 3090EFA 81942F10 32FF184 004009 66724 F1 PG441008 (4027564 DAE6A666 12100 992A7F74 00024831 74000276 3141E990 4557ECC 258154479 B1A07558 D123610E B039023E A5F212F7 DA842E08 12144 66A62E48 701107F1 3301EFA 84081658 64F74AF1 IPG4410A 5328545 97910F4 12145 68A6577 3302FFA 84974179 32F2F187 885C34C2 02090697 CD43EFCF AB00F859 F446344 88F6E29A 928D863C 12276 1990613 D1312582 57A16573 24318L6 04574F1848 12C8792 F608311A 32626934 F15EA055 938470F1 109597C 12326 4000000 0224ACED 00057372 00416F72 672E6170 6163685 2E666966 69227355 63757269 74792E72 65706F73 12326 4000000 00274C 00078700 0000001 7400114 45532F47 743125 (59706F73 69746126 77292E226 12346 06976672 672E6176 6166267 75770707 6656672672 67762674 6405405 9746172 792E226 12346 06976672 672E6176 6166667 757797074 696F6E4D 65746164 61746136 C6904905 9746172 792E526 12346 06976672 672E6176 61636852 E466666 69227365 63757269 74792E72 65706F73 69746172 792E526			PEDODEIC		001 2001 0		201120001	DACE MOR		, our ou			
11722 225656966 69227365 63757269 7479272 65706173 6794672 79225265 70677369 74692672 6726406 65746164 61746136 66944905 97481762 900406772 97225265 53747597 74692676 curity.repository.block.BlockEncryptionMetadata6.I xr Forg.apache.nifi.se 11948 67572697 7492272 55706773 69746772 79225265 76677369 7469676 42797465 73740002 58424000 Metadata.(XN I cipherBytelengthL alg orithmt Ljava/lang/String; [ivBytest [EL 12012 6566579 49667100 76000765 7273696 66710972 09027870 000000101 74001141 45532474 74430274 00248700 07464774 90248700 Netadata.(XN I cipherBytelengthL alg orithmt Ljava/lang/String; [ivBytest [EL KeyIda ~ L versiona ~ N AE/S/GK/N 074ddingur [E T. xp b.A	11748	ØA695616	FDACCB9A	BB1242F5	258EBAA0	207B1D86	2A620000	0224ACED	00057372	00416F72	672E6170	61636865	iV B.% { .*b \$ sr Aorg.apache
11336 G96F6E40 65746144 61746136 CG9246015 G3747269 74792E72 G5706773 G9746F72 792E72 G726773 G9746F72 792E726 G7067369 7467269 G7067369 7467269 G7067369 7467269 G7067369 7467269 G7067369 7467279 G726736 G7067369 G7067369 7467269 G70673657 G7067369 G7067577 S706774 G7067577 S707097 G7067577	11792	2E6E6966	692E7365	63757269	74792E72	65706F73	69746F72	792E626C	6F636B2E	426C6F63	6B456E63	72797074	.nifi.security.repository.block.BlockEncrypt
11880 G3757269 74792E22 G5706F73 G9746F72 792E5265 706F7369 746F729 4F622A65 G372799 746996F6E Curity.repository.Repository.Repository.Bepositor	11836	696F6E4D	65746164	61746136	C69C49D5	97A81F02	00007872	00466F72	672E6170	61636865	2E6E6966	692E7365	ionMetadata6I xr Forg.apache.nifi.se
11924 40657461 64617461 94732858 4EDPTP088 02000549 00016369 70686572 42797465 73740002 SB424000 orithmt Ljava/lag/String;[ivBytest [BL 12012 05665657 49641700 7260267 6776767 7273096 6271007 00027870 0000001740 42027465 73740002 SB424000 keyIdq - L versionq ~ xp t. AES/GCMN 12012 05665657 49641740 7200022 SB424C31 74000276 3141E90 45574C5 25874741 4302744 3402744 3402744 3402744 3402744 3402744 3402744 3402744 3402745 341E90 4557457 3457457 34000276 3141E90 4557457 34745742 3402745 345857453 37490025 S427674 34027444 34027	11880	63757269	74792E72	65706F73	69746F72	792E5265	706F7369	746F7279	4F626A65	6374456E	63727970	74696F6E	curity.repository.RepositoryObjectEncryption
11968 6F726974 656D7400 124C4CA61 76612F6C 616672F 53747269 6E673858 0007067 42797465 73740002 58424C00 reintmt Ljava/Lag/String;[ivytest [BL 12056 6F506164 64596167 75720002 5842ACF3 17F80608 5400200 00078700 0000001 74001141 45532F47 43402F4 opadingur [B T.xp b.A .udf 12140 092A7F74 00024513 74400276 3141E990 4557ECC 5815479 B1A07558 D12306 6476471 F50240 P268675 A400047 002808A1 B6617443790470 2240746 8258679 92808539 7A9310E7 k} yuw.A.A. .cuw.A.A. .c	11924	4D657461	64617461	9F432858	4EDFDF08	02000549	00106369	70686572	42797465	4C656E67	74684C00	09616C67	Metadata.C(XN I cipherByteLengthL alg
12012 05566579 49647100 7E00024C 00077655 7273696F 6271007E 000278700 00000101 74001141 45532F47 4302F4E 69746174 00248173 0Paddingur [B T. xp b.A. udf 12100 992A7F74 000248131 74000278700 00001062 8410008 C4027564 DAEA6666 0Paddingur [B T. xp b.A. udf 12144 66A62EA8 7D1179F5 A39DEF5A B98CF174 4379AE70 322F7183 0A846856 46F744F1 F9C441DA 59289539 74991CF7 %t K1L V.T. xp b.A. vdf 12232 956F76C A4000740 0D2080577 855C42 0200697 OU38FCF A807659 F443464 8876E29A 9280863C <	11968	6F726974	686D7400	124C6A61	76612F6C	616E672F	53747269	6E673B5B	00076976	42797465	73740002	5B424C00	orithmt Ljava/lang/String;[ivBytest [BL
12056 GF566164 64696E67 75720002 SB42ACF3 17F80608 S4E00200 00787000 00001062 C410008 C4027564 DAEGA6666 0*addingur [BT.xpb.A.udf 12140 992A774 00024831 74000276 3141E990 4557ECC2 S8154479 B1A0755B D123610E BD39023E A55212F7 DAB42E0B *.t Kj yZtCy.j Xypd., #a.g.y, #y.g.,, *yp.d.g.y, Yyp.c. 12141 GBA6ZEAB 7D1179F5 A390EF7A 4370AF7D 322FF1138 A4846856 46F74AF1 PG441AD S92B9337 7490EF7A MAD00000 A4AD00A7 0022808A1 B6A11DE9 D4260BC3 00112A3 B08BF9B3 GFAACM4 FFDA7751 3122354 B70113A Cy	12012	056B6579	49647100	7E00024C	00077665	7273696F	6E71007E	00027870	000000D1	74001141	45532F47	434D2F4E	keyIdq ~ L versionq ~ xp .t AES/GCM/N
12100 9922A7F74 0902481 74000276 3141E990 4557ECC2 58154479 B1A07558 D12306 A65721277 DA842ED8 ** K	12056	6F506164	64696E67	75720002	5B42ACF3	17F80608	54E00200	00787000	00001062	8C410D08	C4027564	DAE6A666	oPaddingur [B T. xp b.A .udf
12144 6BA62EAB 7D1179F5 A30BEF5A 898CF124 4379AE7D 32E7F183 0A846856 46F74AF1 F9C441DA 59289539 7A991CF7 k} yZtCy.}2,kVF.JA,Y+.9z. 12184 CF84BF19 0EAFA3C5 1486BF83 CF067577 88534C2 02090697 U043EFCF AB00F859 F4463464 88F6E2A9 92B0865C k} yZtCy.}2,kVF.JA,Y+.9z. 12223 P56EFC6 A4A000A7 00228081 B6A11DE9 D426D8C3 001112A3 808F9893 6AFAC064 FFDAD759 1A223954 B701179A k} yydY92. 12226 1999C031 D13125B2 57A16737 24318CA6 A547FA98 12CB79E2 F603811A 826E0943 F15EAD57 938A70FE 1D05967C f.l(L, Y.F.4, J, dY."9T. q 12326 469746772 792E626C 6F6360842 426C6F63 60456E63 7273074 696F6E4D 6574616 61746136 C694905 97A81F02 xr Forg.apache.nifi.security.repository.Re 12468 000007872 00466772 672E6170 61636865 2E66E966 0927355 63757269 74792E72 65706773 09746772 792E5262 fof23677469 746722792 4F62A65 6374456E 63727307 74696F6E 40657461 641746136 C694969 73664772 792E5265 fof1xr Forg.apache.nifi.security.repository.Re 12450 000106369 70668572 42797465 73740002 58424C00 09568579 49647100 7260024C 00077655 727390F2 fof21672 6726767 7374002 fof1xr Forg.apache.nifi.security.repository.Re 12540 53747269 74672778 850AC77 8450460 744200 95686579 49647100 7260027870 0000081104 145532F47 4340254F4 6F506164 64096567 75720002 58424C69 11267267 7738004000000101 74001141 45532F47 43402F44 6F506126 61682267 13178068 r	12100	992A7F74	00024B31	74000276	3141E990	4557ECC2	58154479	B1A0755B	D123610E	BD39023E	A5F212F7	DAB42ED8	.* t K1t v1AEWX Dyu[.#a .9 >
12188 CF84BF19 0EAFA3CS 148B6F83 CF643F67 88CG422 0299697 CD43EFCF A800F89 F463464 88F6E29A 92B0863C	12144	6BA62EAB	7D1179F5	A39DEF5A	B9BCF174	4379AE7D	32E7F183	ØA846B56	46F74AF1	F9C441DA	592B9539	7A991CE7	k} yZtCy.}2kVF.JA.Y+.9z
12232 9766E76C A4400047 002808A1 B6A110E9 04260823 00112A3 808BF983 6A74064 FFDAD759 1A223954 B701715A, 12276 1999C031 D13125B2 57A16737 24318CA A547FA98 12C676 F603811A 82660943 F15EA57 398A70FE 1D09507C , j, dY "9T. q. 12320 A4D00000 0224ACED 00007872 0446672 6726673 6974672 792E72 6570673 69746772 792E5265 is sr Aorg.apache.nifi.security.repository.Re 12452 70677369 7497279 465626667 7684572 69746772 974525265 is roy.block.BlockEncryptionMetadata.C(XN. I 12452 70667369 74677279 465026667 466547461 97432253 460FDF088 20000740 is roy.block.BlockEncryptionMetadata.C(XN. I 12540 53747269 46677859 7479474 766266167 465347461 97432853 450FDF088 2000749 is roy.block.BlockEncryptionMetadata.C(XN. I 12540 53747269 46677859 7497465 73740002 58424CF3 17880688 20007490 120746124464710 <	12188	CF84BF19	ØEAFA3C5	14B6BF83	CF067577	885C34C2	02090697	CD43EFCF	AB00F859	F4463464	8BF6E29A	92BD863C	uw.\4CY.F4d<
12276 1999C031 D1312582 57A16737 24318CA6 A547FA98 12276E 6903911A 82669043 F15EAD57 938A70FE 1089967C S. sr Aorg.apache.nifi.security.repos 122364 6974672 792E5C6 6753697 7467727 92E5C6 6753697 7467727 92E5C6 67368572 672E6170 61636865 2E6E6966 692E7365 63757269 74792E72 65706F73 69746772 792E5265 ftory.block.BlockEncryptionMetdadta6.I xr Forg.apache.nifi.security.repository.Re 12496 000067329 746F727 92E62A65 63747456 63727970 74696F62 632E74769 74392E72 65706F73 69746F72 792E5265 12496 000166369 76686572 42797465 73740902 58424C00 09616C7 6726974 68607400 12466672 72736976 72739765 727309765 727309765 727309765 727309765 727309765 727309765 727309765 727309765 727309765 727309765 727309765 727309765 727309765 727309765 727309765 72730977 7	12232	9766E76C	A4A0D0A7	0D2808A1	B6A11DE9	D426D8C3	001112A3	8D8BF9B3	6AFAC064	FFDAD759	1A223954	B701719A	.f.l (&jdY "9T. q.
12230 AdD00000 0224ACED 00027372 00416F72 672E6170 61636865 2EEEE966 6927355 63757269 74792E72 65706F73 1:x \$. sr Aorg.apache.nifi.security.repos 12364 69746F72 792E626C 6663682E 42666F63 68456E63 7279707 696F64D 65746164 61746136 669C4905 97A81F02 xr Forg.apache.nifi.security.repos 12480 00007872 0466F72 672E6176 6363685 2EE6E966 6927355 63757269 74792E72 7252E3C6 73727077465 63737269 74792E72 7252E3C6 73729707 74696F62 63726977 7492E72 65706773 69746F72 792E262C 74792E72 725707 74696F67 7470402 584240740 12665677 7470402 58424000 5865774497400 126766572573 697467567 73740002 58424000 5865677442000 5865677442000 58657744000 586476567744000 586476675774000 5864766767 75720002 5842ACF3 17780068 107 × p. 10000276 00002787000 00000107 1011	12276	1999C031	D13125B2	57A16737	24318CA6	A547FA98	12CB79E2	F603811A	826E0943	F15EAD57	938A70FE	1D05967C	1.1%.W.g7\$1Gyn C.^.Wp
12346 69746F72 92262CC 6F63682E 426C6F36 68456E3 72797074 6969F6240 65741614 61746136 C6924905 97A81F02 12428 00007872 00466F72 672651678 61636865 25E66966 692E7365 63757269 74792E72 6570673 69746F72 792E5265 yr Forg.apache.nifi.security.repository.Re 12452 706F7369 74677279 4F626A65 6374456E 63727297 74696F6E 40557461 46417461 97432858 46DF768 20007069 1000001 1000001 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001 100001 1000000000000000000 10000000000000000000000	12320	A4DD0000	0224ACED	00057372	00416F72	672E6170	61636865	2E6E6966	692E7365	63757269	74792E72	65706F73	\$ sr Aorg.apache.nifi.security.repos
12408 00007872 00466F72 672E6170 61636865 2E6E6966 692E7365 63757269 74792E72 65706F73 69746F72 792E5265 xr Forg.apache.nifi.security.repository.Re 12496 00106369 7685727 74677279 4F62A6A5 6374456E 6372797 74696764 19432858 4EDFDF80 20000549 positoryDbjectEncryptionMetadata.CXN I cipher8ytelengthL algorithmt Ljava/lang/ 12540 53747269 6E673B58 00076976 42797465 73740002 SB42400 05686579 49647100 7E00024C 0007765 7273696F 12584 6571007E 00027870 000020010 74001141 45532474 43402F4E 6F596164 46905E76 75720002 S4246067 S17786068 T. xp .t AES/GCM/NoPaddingur [B T. xp .t AES/GCM/NoPaddingur [B T. xp .t AES/GCM/NoPaddingur [S T. xp .t AES/GCM/NoPaddingur [S T. xp .t S	12364	69746F72	792E626C	6F636B2E	426C6F63	6B456E63	72797074	696F6E4D	65746164	61746136	C69C49D5	97A81F02	itory.block.BlockEncryptionMetadata6I
12452 706F7369 746F7229 4F622A95 637247569 74657469 746F7229 4F622A95 63724726 6457461 94617461 9432858 4EDFDF08 02000549 0 <td< td=""><td>12408</td><td>00007872</td><td>00466F72</td><td>672E6170</td><td>61636865</td><td>2E6E6966</td><td>692E7365</td><td>63757269</td><td>74792E72</td><td>65706F73</td><td>69746F72</td><td>792E5265</td><td>xr Forg.apache.nifi.security.repository.Re</td></td<>	12408	00007872	00466F72	672E6170	61636865	2E6E6966	692E7365	63757269	74792E72	65706F73	69746F72	792E5265	xr Forg.apache.nifi.security.repository.Re
12496 0016359 70686522 42797465 46620400 09616C67 67726974 68070490 1246A61 76612F6C 615672F 11	12452	706F7369	746F7279	4F626A65	6374456E	63727970	74696F6E	4D657461	64617461	9F432858	4EDFDF08	02000549	positoryObjectEncryptionMetadata.C(XN I
12540 \$3747269 6E673858 00076976 42797465 73740002 \$8424000 05686579 49647100 7E00024C 00077665 7273696F String; [ivBytest [BL keyIdq ~ L versio 12584 6E71007E 000027870 00002001 74001141 45532447 43402F4E 6F506164 64696EC7 75720002 \$842ACF3 17F80608 nq ~ xp .t AES/GCM/NoPaddingur [B 12628 54600200 000787000 00000106 8AbF06C7 E564F65 145182E A8380E74 00024831 74000276 3161CB12 5454606F T. xpd.e E8.t Klt vla. TT o 12672 24A497A1 FB67CF78 3E0A4CF7 9C2E4EE 2A7438C2 541414 42788649 F506E026 GE8825E 1370428 5g.x6.L*t8.P. BI. Nmakrim.(.^17 + 12760 36A9219C GE335DCA AESDDZEA 79069A62 2222D571 88030580 9009E5CC 32870411 4173939A E28DF9AD emso&.7 ?.T(> As 6%!.b.]]y.b", q.m.UCI 4\pyk2((.R= {, 1,, n, makrim.(, 1/ + , pk2 12848 480EE409 98926647 108723F0 3744280	12496	00106369	70686572	42797465	4C656E67	74684C00	Ø9616C67	6F726974	686D7400	124C6A61	76612F6C	616E672F	cipherByteLengthL algorithmt Ljava/lang/
12584 6E71007E 00027870 00000001 74001141 45532F47 43402F4E 6F5906164 64596E67 75720002 5842ACF3 17780608 nq ~ xp .t. AES/GCM/NoPaddingur [B 12626 54600200 00787000 0000010C6 8ADF06C7 E564EF65 145312E4 AB38DE74 00024831 7400276 3161CB12 5454060F T. xpd.e Es.t K1t v1a. TT`o 12672 244397A1 FB5C/F78 8CDAUCT9 9CC2E4EB 2A7438C2 50F415AE 42F88649 FS06026 61682260 DE28655 13792E3 5g.x6.L*t8.PBI. Mack'm.(_^17 + 12716 B1206560 388407BD CC224EB 2A7438C2 S0F415AE 42F88649 FS060206 G1682260 DE286255 13792E3 Sg.x6.L*t8.PBI. Mack'm.(_^17 + 12760 36A92196 G2835DCA AE5DD2EA 79069A62 222D571 B85D0755 BE43913 345C0EA2 9270796B 32DDFD28 2894523D G+	12540	53747269	6E673B5B	00076976	42797465	73740002	5B424C00	056B6579	49647100	7E00024C	00077665	7273696F	String;[ivBytest [BL keyIdq ~ L versio
12628 S4E00200 000787000 000010C6 8ADF06C7 E564F65 1451B2E AB38DE74 00024B31 74000276 3161CB12 5454606F T. xp d.e E .8.t Kl t v1a TT'o 12672 24A497A1 FB67CF78 36DA4CF7 9CC2E4EB 2A7438C2 50F415AE 42F88649 F5066D26 616B226D CB288C5E 31370E2B 12716 B12060 36A9219C 62835DCA AESDD2EA 7003A92ED 5484E94 2883009500905CC 32870411 4133939A E280F9A0 640 64(+,b,-],.],.y, b"", q,m.U.CI 4, vpk2(C(.* 12804 04907B3B A5C0214A F6E5098A 70384022 058A5EB 8665511F 3F5E1914 DC591E24 FF02705 64(+,b,-],], y, b"", q,m.U.CI 4, vpk2(C(.* 12848 404907B3B A5C0214A F6E5098A 70384022 058A5E30 49847F 12848 404907B32 57246042 98256647 108723F0 3744A280 6524FE32 05A42530 49847F 12848 404907B38 A5C0214A F6E5098A 70384022	12584	6E71007E	00027870	000000D1	74001141	45532F47	434D2F4E	6F506164	64696E67	75720002	5B42ACF3	17F80608	nq ~ xp .t AES/GCM/NoPaddingur [B
12672 24A497A1 FB67CF78 36DA4CF7 9CC2E4EB 2A7438C2 50F415AE 42FB8649 F5066D26 616B226D CB288C5E 31370E2B 12760 81206560 388407BD CA28585 B66F26BA 370A3FED 5488E94 28830B0 9009ESCC 3870411 4173939A E28DF9AD 12760 36A9219C 628350CA AESDD2EA 79069A62 22220571 B860D755 BE434913 345C0EA2 9270796B 320DF028 2894523D 6%!	12628	54E00200	00787000	000010C6	8ADF06C7	E564EF65	1E451B2E	AB38DE74	00024B31	74000276	3161CB12	5454606F	T. xpd.e E8.t K1t v1a. TT`o
12716 B1206560 388407B0 CCA28585 B66F26BA 370A3FED 5488E894 28839D80 9009E5CC 3E870411 4173939A E28DF9AD .	12672	24A497A1	FB67CF78	36DA4CF7	9CC2E4EB	2A7438C2	50F415AE	42FB8649	F5066D26	616B226D	CB288C5E	31370E2B	\$g.x6.L*t8.PBI. m&ak"m.(.^17 +
12260 36A9219C 62835DCA AE5D02EA 79069A62 2222D571 B86D0755 BE434913 345C0EA2 92707968 32DDFD28 2894523D 60(+, b_, 1, j_, v_, b''', q, m, U, CI \ pyk2((. k_, 1, j_, v_, b''', q, m, U, CI \ pyk2((. k_, 1, j_, v_, b''', q, m, U, CI \ pyk2((. k_, 1, j_, v_, b''', q, m, U, CI \ pyk2((. k_, 1, j_, v_, b''', q, m, U, CI \ pyk2((. k_, 1, j_, v_, b''', q, m, U, CI \ pyk2((. k_, 1, j_, v_, b''', q, m, U, CI \ pyk2((. k_, 1, j_, v_, b''', q, m, U, CI \ pyk2(I, v_, 1, j_, v_, b''', q, m, U, CI \ pyk2(I, v_, 1, j_, v_, b''', q, m, U, CI \ pyk2(I, v_, 1, j_, v_, b''', q, m, U, CI \ pyk2(I, v_, 1, j_, v_, b''', q, m, U, CI \ pyk2(I, v_, 1, j_, v_, b''', q, m, U, CI \ pyk2(I, v_, 1, j_, v_, b''', q, m, U, CI \ pyk2(I, v_, 1, j_, v_, b''', q, m, U, CI \ pyk2(I, v_, 1, j_, v_, b''', q, m, U, CI \ pyk2(I, v_, 1, j_, v_, b''', q, m, U, CI \ pyk2(I, v_, 1, j_, v_, b''', q, m, U, CI \ pyk2(I, v_, 1, j_, v_, b''', q, m, U, CI \ pyk2(I, v_, 1, j_, v_, b''', q, m, U, CI \ pyk2(I, v_, 1, j_, v_, v_, v_, v_, v_, v_, v_, v_, v_, v	12716	B120656D	388407BD	CCA28585	B66F26BA	370A3FED	5488EB94	28839DB0	9D09E5CC	3E870411	4173939A	E28DF9AD	. em8o&.7 ?.T(>. As
12804 049D7B3B A5C0214A F6E5098A 703B4022 0F8BFBE1 BCC9C35B 4665511F 3F5E1914 DC591E24 FE75979C E7F02705 . {; J p;@" [FeQ ?^ Y \$.u' 12848 4B0EE409 9B926647 1D8723F0 3744A280 6824FE32 05AA2630 498A7F Signed Int ○ [ie, hex] 0xEDAC	12760	36A9219C	62835DCA	AE5DD2EA	79069A62	2222D571	B86DD755	BE434913	345C0EA2	9270796B	32DDFD28	2894523D	60!.b.]]y .b"".q.m.U.CI 4\pyk2((.R=
12848 480EE409 9B926647 1D8723F0 3744A280 6824FE32 05AA2630 498A7F K fG .#.7Dh\$.2 .&01. Signed Int ○ [e, hex] 0xEDAC	12804	049D7B3B	A5C0214A	F6E5098A	703B4022	ØF8BFBE1	BCC9C35B	4665511F	3F5E1914	DC591E24	FE75979C	E7F02705	.{;!Jp;@"[FeQ ?^ .Y \$.u'
Signed Int ♦ (le, hex) 0xEDAC	12848	4B0EE409	9B926647	1D8723F0	3744A280	6824FE32	05AA2630	498A7F					KfG .#.7Dh\$.2 .&0I.
	Signed	Int 🗘 le, h	ex 0xEDA	0									
uxz bytes selected at orrset 0x1CBE out of 0x324B bytes								0x2 bytes s	elected at offse	t 0x1CBE out o	f 0x324B bytes		

On flowfile record read, the process is reversed. The encryption metadata (RepositoryObjectEncryptionMetadata) is parsed and used to decrypt the serialized bytes, which are then deserialized into a DataInputStream object.

During swaps and recoveries, the flowfile records are descrialized and reserialized, so if the active key has been changed, the flowfile records will be re-encrypted with the new active key.

Within the NiFi UI/API, there is no detectable difference between an encrypted and unencrypted flowfile repository. All framework interactions with flowfiles work as expected with no change to the process.

Potential Issues

- Switching between unencrypted and encrypted repositories
 - If a user has an existing write-ahead repository (WriteAheadFlowFileRepository) that is not encrypted (uses the SequentialAccessWriteAheadLog) and switches their configuration to use an encrypted repository, the application handles this and all flowfile records will be recovered on startup. Future writes (including reserialization of these same flowfiles) will be encrypted. If a user switches from an encrypted repository to an unencrypted repository, the flowfiles cannot be recovered, and it is recommended to delete the existing flowfile repository before switching in this direction. Automatic roll-over is a future effort (https://issues.apache.org/jira/browse/NIFI-6994) but NiFi is not intended for long-term storage of flowfile records so the impact should be minimal. There are two scenarios for roll-over:
 - Encrypted # unencrypted if the previous repository implementation was encrypted, these records should be handled seamlessly as long as the key provider available still has the keys used to encrypt the claims (see *Key Rotation*.)
 - Unencrypted # encrypted currently handled seamlesssly for SequentialAccessWriteAheadLog but there are other initial implementations which could be handled
 - There is also a future effort to provide a standalone tool in NiFi Toolkit to encrypt/decrypt an existing flowfile repository to make the transition easier. The translation process could take a long time depending on the size of the existing repository, and being able to perform this task outside of application startup would be valuable (https://issues.apache.org/jira/browse/NIFI-6994).
- Multiple repositories No additional effort or testing has been applied to multiple repositories at this time. Current implementations of the flowfile repository allow only for one repository, though it can reside across multiple volumes and partitions. It is possible/likely issues will occur with repositories on different physical devices. There is no option to provide a heterogenous environment (i.e. one encrypted, one plaintext partition/directory).

• Corruption - when a disk is filled or corrupted, there have been reported issues with the repository becoming corrupted and recovery steps are necessary. This is likely to continue to be an issue with the encrypted repository, although still limited in scope to individual records (i.e. an entire repository file won't be irrecoverable due to the encryption). It is important for the continued operation of NiFi to ensure that the disk storing the flowfile repository does not run out of available space.

Experimental Warning

While all Apache licensed code is provided "on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied" (see https://www.apache.org/licenses/LICENSE-2.0), some features of Apache NiFi may be marked experimental. Experimental features may:

- · have undergone less extensive testing than is normal for standard NiFi features
- interact with unstable external dependencies
- be subject to change (any exposed APIs should not be considered covered under the minor release backward compatibility guidelines of https://semver.org)
- potentially cause data loss
- not be directly supported by the community in the event issues arise

Every attempt is made to provide more detailed and specific information around the nature of the experimental warning on a per-feature basis. Questions around specific experimental features should be directed to the dev@nifi.apache.org.

Other Management Features

In addition to the Summary Page, Data Provenance Page, Template Management Page, and Bulletin Board Page, there are other tools in the Global Menu (see NiFi User Interface) that are useful to the DFM. Select Flow Configuration History to view all the changes that have been made to the dataflow. The history can aid in troubleshooting, such as if a recent change to the dataflow has caused a problem and needs to be fixed. The DFM can see what changes have been made and adjust the flow as needed to fix the problem. While NiFi does not have an "undo" feature, the DFM can make new changes to the dataflow that will fix the problem.

Two other tools in the Global Menu are Controller Settings and Users. The Controller Settings page provides the ability to change the name of the NiFi instance, add comments describing the NiFi instance, and set the maximum number of threads that are available to the application. It also provides tabs where DFMs may add and configure Controller Services and Reporting Tasks. The Users page is used to manage user access, which is described in the System Administrator's Guide.