Cloudera Flow Management 4.10.0

Cloudera Flow Management Release Notes

Date published: 2019-06-26
Date modified: 2025-05-22

https://docs.cloudera.com/

Legal Notice

Contents

What's new in Cloudera Flow Management 4.10.0

Explore the new features and improvements in Cloudera Flow Management and learn how they can modernize your data pipeline development and operations.

Cloudera Flow Management 4.10.0 marks the General Availability (GA) release of Apache NiFi 2 within Cloudera's data platform.

This release introduces a broad set of NiFi 2 changes along with Cloudera-exclusive capabilities and improvements, providing an enterprise-ready foundation for building, operating, and scaling your data pipelines. Switching to NiFi 2 in your data flow development and operation is not just a technical update, it is a modernization of the whole data flow experience, offering greater performance, flexibility, and security.

New features and changes

The following sections highlight the most important changes introduced with NiFi 2 and Cloudera Flow Management 4.10.0.

Native Python 3 support

NiFi 2 introduces a powerful Python API, enabling you to develop processors, controller services, and reporting tasks directly in Python. This allows you to embed Python-based orchestration and data manipulation into your flows without relying on external scripting.

AI & GenAI components

NiFi 2 provides a framework for building A-enabledI processors. Cloudera extends this with production-grade GenAI components, supporting advanced AI-driven workflows.

Flow Analysis Rules

A new, built-in rules engine enables real-time data quality validation within your flows. You can define rules to detect schema violations, missing fields, or implement custom business logic, all without additional coding.

Enhanced authentication

NiFi 2 removes legacy Kerberos configuration methods and enforces the use of Kerberos UserService. Authentication and encryption are now stricter and more secure by default.

Parameterization enhancement

Global variables have been replaced by a granular parameter framework, improving flow modularity and manageability.

Redesigned user interface

The new NiFi 2 UI is faster, cleaner, and more intuitive. It supports modular flow design with reusable components, speeding up development and allows for the centralization of design patterns in large projects.

Removed components

Over 140 components have been removed and must be replaced in NiFi 2. For more information on breaking changes between NiFi 1 and 2, see Behavioral changes.

Readded components

Cloudera Flow Management 4.10.0 restores support for several NiFi components that were deprecated in Apache NiFi 2. These components have been re-added to maintain compatibility and support key use cases. For more information, see Reintroduced NiFi components.

Migration tool

Cloudera provides a Flow Management Migration Tool that helps you replace deprecated processors, update configurations, and handle breaking changes automatically in your data flows. For more information, see Cloudera Flow Management Migration Tool.

Other improvements

NiFi 2 brings significant performance enhancements, especially important for high-volume workloads. It expands connectivity options both for source and target systems, simplifying to connect your data pipelines in hybrid environments. With richer native integration features, you can reduce reliance on custom processors. NiFi 2 also improves developer productivity with enhanced SDKs for Python and Java, enabling faster development and more efficient flow management.

New NiFi components

Cloudera Flow Management 4.10.0 introduces several new NiFi components to support broader integration and processing capabilities.

New processors:

- GetBoxFileCollaborators
- ExecuteSparkInteractive
- GetBoxGroupMembers
- ConsumeBoxEnterpriseEvents
- SawmillTransformRecord
- PutSolrRecord
- CaptureChangeDebeziumMongoDB
- SawmillTransformJSON
- GetSolr
- GetS3ObjectTags
- FetchBoxFileRepresentation
- PutSolrContentStream
- ListBoxFileInfo
- QuerySolr
- ListenBeats
- ConsumeBoxEvents
- FetchBoxFileInfo

Python processors:

- New
  - PromptClaude
  - TokenCount
  - PromptAzureOpenAI
  - PromptOpenAI
- Renamed
  - Bedrock renamed to PromptBedrock

New controller services:

- ClouderaEncodedSchemaReferenceReader
- ClouderaEncodedSchemaReferenceWriter
- PhoenixThickConnectionPool
- PhoenixThinConnectionPool
- ClouderaAttributeSchemaReferenceWriter
- DeveloperBoxClientService
- RESTCatalogService

- ClouderaAttributeSchemaReferenceReader
- LivySessionController
- PEMEncodedSSLContextProvider
- StandardDatabaseDialectService

New parameter provider:

- PropertiesFileParameterProvider

Flow Analysis Rules:

- New
  - RequireMergeBeforePutIceberg
  - RestrictFlowFileExpiration
- Replaced
  - RestrictBackpressure replaced with RestrictBackpressureSettings

For the full list of supported NiFi components, see the Support Matrix.

Reintroduced NiFi components

Cloudera Flow Management 4.10.0 restores support for several NiFi components that were deprecated upstream in Apache NiFi 2.

Processors:

- PutSolrContentStream
- PutSolrRecord
- QuerySolr
- ExecuteSparkInteractive

Controller service:

- LivySessionController

Breaking changes between NiFi 1 and NiFi 2

Cloudera Flow Management 4.10.0 incorporates key changes to maintain compatibility with Cloudera's ecosystem while aligning with the open-source NiFi 2 project.So as part of the transition from NiFi 1 to NiFi 2, Cloudera has moved a large number of components downstream that were removed from the open-source Apache NiFi project. These adjustments aim to provide better support for Cloudera customers by aligning with specific Cloudera technologies and minimizing disruptive changes.

For example:

- Hive components: NiFi in Cloudera Flow Management includes Hive components built against Cloudera's version of Hive, which has diverged from Apache Hive, ensuring compatibility with Cloudera's ecosystem.
- Kafka components: Cloudera has retained Kafka 2.6 components in Cloudera Flow Management to facilitate smoother transitions from legacy Kafka components to the new Kafka architecture based on controller services, reducing the risk of breaking changes.
- Jython scripting: Jython scripting support remains available for certain scripting components, despite being removed upstream.

For more information on these breaking changes, see Behavioral changes.

Upgrade and migration options

There is no supported in-place upgrade path from Cloudera Flow Management versions powered by NiFi 1 (2.1.7 and earlier) to Cloudera Flow Management 4.10.0.

Cloudera provides a Migration Tool that automates complex, repetitive, and error-prone manual tasks in updating flow configurations, reducing manual effort and ensuring compatibility with NiFi 2 features.

This Cloudera Flow Management Migration Tool simplifies the transition by:

- Replacing removed processors
- Converting variable-based configurations to parameters
- Reconfiguring flows to use new controller services
- Converting older templates into flow definitions
- Adapting to security, data type, and API changes introduced in NiFi 2

Important:

To transition from Cloudera Flow Management versions powered by NiFi 1 to Cloudera Flow Management 4.10.0 and use the Migration Tool for migrating your flows, you must be on Cloudera Flow Management 2.1.7. For version details, see the Component versions page.

For more information, see the Cloudera Flow Management Migration Tool documentation.

If you previously installed Cloudera Flow Management 4.0.0 [Technical Preview], you can upgrade to 4.10.0 directly. For instructions, see Upgrading from Cloudera Flow Management 4.0.0 to 4.10.0,

Installation

To install Cloudera Flow Management 4.10.0 from scratch, follow the Cloudera Flow Management installation workflow.

Support matrix

Review the support matrix before you start installing Cloudera Flow Management.

Component versions

Review the Cloudera Flow Management component versions for compatibility with other applications.

Note: NiFi is compatible with the version of NiFi Registry bundled with your Cloudera Flow Management release as well as any later version.

Cloudera Flow Management 4.10.0

- Apache NiFi 2.3.0.4.10.0.0
- Apache NiFi Registry 2.3.0.4.10.0.0

Cloudera Flow Management 4.0.0 [Technical Preview]

- Apache NiFi 2.0.0.4.0.0.0
- Apache NiFi Registry 2.0.0.4.0.0.0

Cloudera Flow Management 2.1.7.2000 (SP2)

- Apache NiFi 1.28.1.2.1.7.2000
- Apache NiFi Registry 1.28.1.2.1.7.2000

Cloudera Flow Management 2.1.7.1000 (SP1)

- Apache NiFi 1.26.0.2.1.7.1000
- Apache NiFi Registry 1.26.0.2.1.7.1000

Cloudera Flow Management 2.1.7

- Apache NiFi 1.26.0.2.1.7.0
- Apache NiFi Registry 1.26.0.2.1.7.0

Cloudera Flow Management 2.1.6.1000 (SP1)

- Apache NiFi 1.23.1.2.1.6.1000
- Apache NiFi Registry 1.23.1.2.1.6.1000

Cloudera Flow Management 2.1.6

- Apache NiFi 1.23.1.2.1.6.0
- Apache NiFi Registry 1.23.1.2.1.6.0

Cloudera Flow Management 2.1.5.1000 (SP1)

- Apache NiFi 1.18.0.2.1.5.1000
- Apache NiFi Registry 1.18.0.2.1.5.1000

Cloudera Flow Management 2.1.5

- Apache NiFi 1.18.0.2.1.5.0
- Apache NiFi Registry 1.18.0.2.1.5.0

Cloudera Flow Management 2.1.4.1000 (SP1)

- Apache NiFi 1.16.0.2.1.4.1000
- Apache NiFi Registry 1.16.0.2.1.4.1000

Cloudera Flow Management 2.1.4

- Apache NiFi 1.16.0.2.1.4.0
- Apache NiFi Registry 1.16.0.2.1.4.0

Cloudera Flow Management 2.1.3

- Apache NiFi 1.15.2.2.1.3.0
- Apache NiFi Registry 1.15.2.2.1.3.0

Note: Apache NiFi and Apache NiFi Registry versions are unified in the 1.15.x release.

Cloudera Flow Management 2.1.2

- Apache NiFi 1.13.2.2.1.2.0
- Apache NiFi Registry 0.8.0.2.1.2.0

Cloudera Flow Management 2.1.1

- Apache NiFi 1.13.2.2.1.1.0
- Apache NiFi Registry 0.8.0.2.1.1.0

Cloudera Flow Management 2.0.4

- Apache NiFi 1.11.4
- Apache NiFi Registry 0.6.0

Cloudera Flow Management 2.0.1

- Apache NiFi 1.11.4
- Apache NiFi Registry 0.6.0

System requirements

Review the system requirements before getting started with installing Cloudera Flow Management 4.10.0.

Supported Cloudera versions

Cloudera Flow Management 4.10.0 supports the following versions of Cloudera Base on premises:

- Cloudera 7.3.1

Supported JAVA Development Kits (JDK)

Cloudera Flow Management 4.10.0 requires a minimum of JDK 21 for proper functionality. Ensure your environment meets this requirement before installation.

Other system requirements

ZooKeeper

You need to install the ZooKeeper service included with your Cloudera Base on premises cluster.

Python

- Minimum requirement: Python 3.11
- Recommended version: Python 3.12

Number of cores

- Minimum: Four cores per NiFi node are required for Cloudera support.
- Recommended: Eight cores per NiFi node, which typically provides an optimal starting point for most common use cases.

Supported operating systems

Review the list of operating systems supported in Cloudera Flow Management 4.10.0.

Operating system	Versions
RHEL	• 8.8 • 8.10 • 9.1 • 9.2 • 9.4
Oracle	• 8.8
Ubuntu	• 20.04 • 22.04
Windows	• 10 • Server 2016 • Server 2019 • Server 2022 • Server 2025

Note:

NiFi on Windows is only supported in standalone mode, not managed by Cloudera Manager or as part of a Cloudera cluster, and as a single instance installation. Clustering NiFi on Windows is not supported.

NiFi Registry is not supported on Windows.

Supported NiFi Registry databases

Review the list of databases supported by NiFi Registry in Cloudera Flow Management 4.10.0.

- H2
- PostgreSQL 10.x
- PostgreSQL 11.x
- PostgreSQL 12.x
- PostgreSQL 13.x
- PostgreSQL 14.x
- MySQL 8.x

Supported NiFi processors

Review the list of Apache NiFi processors supported in Cloudera Flow Management 4.10.0.

Cloudera Flow Management is based on Apache NiFi and includes a set of processors, most of which are supported by Cloudera. To ensure optimal performance and reliable support, it is crucial to use only supported processors and avoid deploying unsupported ones in production environments.

Additional processors are developed and tested by the community but are not officially supported by Cloudera. Processors may be excluded for various reasons, including insufficient reliability, incomplete test coverage, community declaration of non-production readiness, or deviations from Cloudera best practices.

By adhering to the above guidelines, you can maintain stable and reliable workflows in your production environments.

AttributesToCSV	GetBoxGroupMembers
AttributesToJSON	GetCouchbaseKey1
CalculateParquetOffsets	GetElasticsearch
CalculateParquetRowGroupOffsets	GetFile
CalculateRecordStats	GetFTP
CaptureChangeDebeziumDB2 [Technial Preview]	GetGcpVisionAnnotateFilesOperationStatus
CaptureChangeDebeziumMongoDB [Technial Preview]	GetGcpVisionAnnotateImagesOperationStatus
CaptureChangeDebeziumMySQL [Technial Preview]	GetHBase
CaptureChangeDebeziumOracle	GetHDFS
CaptureChangeDebeziumPostgreSQL	GetHDFSFileInfo
CaptureChangeDebeziumSQLServer [Technial Preview]	GetHDFSSequenceFile
CaptureChangeMySQL	GetHubSpot
ChunkDocument	GetJiraIssue
CompressContent1, 2	GetMongoRecord
ConnectWebSocket	GetS3ObjectTags
ConsumeAMQP	GetSFTP
ConsumeAzureEventHub	GetShopify

ConsumeBoxEnterpriseEvents	GetSNMP
ConsumeBoxEvents	GetSnowflakeIngestStatus
ConsumeElasticsearch	GetSolr
ConsumeGCPubSub	GetSplunk
ConsumeGCPubSubLite	GetSQS
ConsumeJMS	GetWorkdayReport
ConsumeKafka_2_6	GetZendesk
ConsumeKafka2CDP	HandleHttpRequest
ConsumeKafka2RecordCDP	HandleHttpResponse
ConsumeKafkaRecord_2_6	IdentifyMimeType
ConsumeKinesisStream	InvokeAWSGatewayApi
ConsumeMQTT1	InvokeGRPC
ConsumePLC [Technial Preview]	InvokeHTTP
ConsumeSlack	InvokeScriptedProcessor
ConsumeTwitter	JoinEnrichment
ConsumeWindowsEventLog	JoltTransformJSON
ControlRate	JoltTransformRecord
ConvertAvroToJSON	JSLTTransformJSON
ConvertAvroToParquet	JsonQueryElasticsearch
ConvertCharacterSet	ListAzureBlobStorage_v12
ConvertJSONToSQL	ListAzureDataLakeStorage
ConvertProtobuf	ListBoxFile
ConvertRecord	ListBoxFileInfo
CopyAzureBlobStorage_v12	ListCDPObjectStore
CountText	ListDatabaseTables
CreateHadoopSequenceFile	ListDropbox
CryptographicHashContent	ListenBeats
DecryptContent	ListenFTP
DecryptContentAge	ListenGRPC
DecryptContentCompatibility	ListenHTTP
DecryptContentPGP	ListenNetFlow
DeduplicateRecord	ListenOTLP
DeleteAzureBlobStorage_v12	ListenRELP
DeleteAzureDataLakeStorage	ListenSlack
DeleteByQueryElasticsearch	ListenSyslog
DeleteCDPObjectStore	ListenTCP
DeleteDynamoDB	ListenTCPRecord
DeleteGCSObject	ListenTrapSNMP
DeleteGridFS	ListenUDP
DeleteHBaseCells	ListenUDPRecord

DeleteHBaseRow	ListenWebSocket
DeleteHDFS	ListFile
DeleteS3Object	ListFTP
DeleteSQS	ListGCSBucket
DetectDuplicate	ListGoogleDrive
DistributeLoad	ListHDFS
DuplicateFlowFile	ListS3
EncodeContent	ListSFTP
EncryptContentAge	ListSmb
EncryptContentPGP	LogAttribute
EnforceOrder	LogMessage
EvaluateJsonPath	LookupAttribute
EvaluateXPath	LookupRecord
EvaluateXQuery	MergeContent
ExecuteGroovyScript	MergeRecord1
ExecuteProcess	ModifyCompression
ExecuteScript	MonitorActivity
ExecuteSQL	MoveAzureDataLakeStorage
ExecuteSQLRecord	MoveHDFS
ExecuteStateless1, 2	Notify
ExecuteStreamCommand	PackageFlowFile
ExtractAvroMetadata	PaginatedJsonQueryElasticsearch
ExtractGrok	ParseCEF1
ExtractHL7Attributes	ParseDocument
ExtractImageMetadata	ParseEvtx
ExtractRecordSchema	ParseSyslog
ExtractText	PartitionRecord
FetchAzureBlobStorage_v12	PromptAzureOpenAI
FetchAzureDataLakeStorage	PromptBedrock
FetchBoxFile	PromptChatGPT
FetchBoxFileInfo	PromptClaude
FetchBoxFileRepresentation	PromptOpenAI
FetchCDPObjectStore	PublishAMQP
FetchDistributedMapCache	PublishGCPubSub1
FetchDropbox	PublishGCPubSubLite1
FetchFile	PublishJMS1
FetchFTP	PublishKafka_2_6
FetchGCSObject	PublishKafka2CDP
FetchGoogleDrive	PublishKafka2RecordCDP
FetchGridFS	PublishKafkaRecord_2_6

FetchHBaseRow	PublishMQTT
FetchHDFS	PublishSlack
FetchParquet	PutAccumuloRecord1
FetchPLC [Technial Preview]	PutAzureBlobStorage_v12
FetchS3Object	PutAzureCosmosDBRecord
FetchSFTP	PutAzureDataLakeStorage1
FetchSmb	PutAzureEventHub
FilterAttribute	PutAzureQueueStorage_v12
FlattenJson	PutBigQuery
ForkEnrichment	PutBoxFile
ForkRecord	PutCassandraQL1
GenerateFlowFile	PutCassandraRecord1
GenerateRecord	PutCDPObjectStore
GenerateTableFetch	PutChroma
GeoEnrichIP	PutClouderaHiveQL
GeoEnrichIPRecord	PutClouderaHiveStreaming
GeohashRecord	PutClouderaORC
GetAsanaObject	PutCloudWatchMetric
GetAwsPollyJobStatus	PutCouchbaseKey
GetAwsTextractJobStatus	PutDatabaseRecord1
GetAwsTranscribeJobStatus	PutDistributedMapCache
GetAwsTranslateJobStatus	PutDropbox
GetAzureEventHub	PutDynamoDB
GetAzureQueueStorage_v12	PutDynamoDBRecord
GetBoxFileCollaborators	

Footnotes

- 1 – indicates a memory intensive processor
- 2 – indicates a CPU intensive processor

Related Information

Supported NiFi controller services
Supported NiFi reporting tasks
Supported NiFi parameter providers
Supported NiFi flow analysis rules
Supported NiFi flow registry clients

Supported NiFi controller services

Review the list of Apache NiFi controller services supported in Cloudera Flow Management 4.10.0.

Cloudera Flow Management is based on Apache NiFi and includes a set of controller services, most of which are supported by Cloudera. To ensure optimal performance and reliable support, it is crucial to use only supported controller services and avoid deploying unsupported ones in production environments.

Additional controller services are developed and tested by the community but are not officially supported by Cloudera. Controller services may be excluded for various reasons, including insufficient reliability, incomplete test coverage, community declaration of non-production readiness, or deviations from Cloudera best practices.

By adhering to the above guidelines, you can maintain stable and reliable workflows in your production environments.

AccumuloService	IPLookupService
ActiveMQJMSConnectionFactoryProvider	JASN1Reader
ADLSCredentialsControllerService	JiraRecordSink
ADLSCredentialsControllerServiceLookup	JMSConnectionFactoryProvider
ADLSIDBrokerCloudCredentialsProviderControllerService	JndiJmsConnectionFactoryProvider
AmazonGlueSchemaRegistry	JsonConfigBasedBoxClientService
ApicurioSchemaRegistry	JsonPathReader
AvroReader	JsonRecordSetWriter
AvroRecordSetWriter	JsonTreeReader
AvroSchemaRegistry	KafkaRecordSink_2_6
AWSCredentialsProviderControllerService	KerberosKeytabUserService
AWSIDBrokerCloudCredentialsProviderControllerService	KerberosPasswordUserService
AzureBlobIDBrokerCloudCredentialsProviderControllerService	KerberosTicketCacheUserService
AzureCosmosDBClientService	KuduLookupService
AzureEventHubRecordSink	LoggingRecordSink
AzureServiceBusJMSConnectionFactoryProvider	MongoDBControllerService
AzureStorageCredentialsControllerService_v12	MongoDBLookupService
AzureStorageCredentialsControllerServiceLookup_v12	ParquetReader
CassandraDistributedMapCache	ParquetRecordSetWriter
CassandraSessionProvider	PEMEncodedSSLContextProvider
CdpCredentialsProviderControllerService	PhoenixThickConnectionPool
CdpOauth2AccessTokenProviderControllerService	PhoenixThinConnectionPool
CEFReader	PostgreSQLConnectionPool
CiscoEmblemSyslogMessageReader	PrometheusRecordSink
ClouderaAttributeSchemaReferenceReader	ProxyPLC4XConnectionPool [Technical Previe
ClouderaAttributeSchemaReferenceWriter	RabbitMQJMSConnectionFactoryProvider
ClouderaEncodedSchemaReferenceReader	ReaderLookup
ClouderaEncodedSchemaReferenceWriter	RecordSetWriterLookup
ClouderaHiveConnectionPool	RecordSinkServiceLookup
ClouderaSchemaRegistry	RedisConnectionPoolService
CMLLookupService	RedisDistributedMapCacheClientService
ConfluentEncodedSchemaReferenceReader	RedshiftConnectionPool
ConfluentEncodedSchemaReferenceWriter	RESTCatalogService
ConfluentSchemaRegistry	RestLookupService
CouchbaseClusterService	ScriptedLookupService
CouchbaseKeyValueLookupService	ScriptedReader

CouchbaseMapCacheClient	ScriptedRecordSetWriter
CouchbaseRecordLookupService	ScriptedRecordSink
CSVReader	SimpleDatabaseLookupService
CSVRecordLookupService	SimpleKeyValueLookupService
CSVRecordSetWriter	SimpleRedisDistributedMapCacheClientServic
DatabaseRecordLookupService	SimpleScriptedLookupService
DatabaseRecordSink	SiteToSiteReportingRecordSink
DatabaseTableSchemaRegistry	SlackRecordSink
DeveloperBoxClientService	SmbjClientProviderService
DBCPConnectionPool	SnowflakeComputingConnectionPool
DBCPConnectionPoolLookup	StandardAsanaClientProviderService
DistributedMapCacheClientService	StandardAzureCredentialsControllerService
DistributedMapCacheLookupService	StandardDatabaseDialectService
DistributedMapCacheServer	StandardDropboxCredentialService
DistributedSetCacheClientService	StandardFileResourceService
DistributedSetCacheServer	StandardHashiCorpVaultClientService
EBCDICRecordReader [Technical Preview]	StandardHttpContextMap
ElasticSearchClientServiceImpl	StandardJsonSchemaRegistry [Technical Previe
ElasticSearchLookupService	StandardOauth2AccessTokenProvider
ElasticSearchStringLookupService	StandardPGPPrivateKeyService
EmailRecordSink	StandardPGPPublicKeyService
EmbeddedHazelcastCacheManager	StandardPLC4XConnectionPool [Technical Pre
ExcelReader	StandardPrivateKeyService
ExternalHazelcastCacheManager	StandardProxyConfigurationService
FreeFormTextRecordSetWriter	StandardRestrictedSSLContextService
GCPCredentialsControllerService	StandardS3EncryptionService
GCSFileResourceService	StandardSnowflakeIngestManagerProviderServ
GenericPLC4XConnectionPool [Technical Preview]	StandardSSLContextService
GrokReader	StandardWebClientServiceProvider
HadoopCatalogService	Syslog5424Reader
HadoopDBCPConnectionPool	SyslogReader
HazelcastMapCacheClient	UDPEventRecordSink
HBase_2_ClientMapCacheService	VolatileSchemaCache
HBase_2_ClientService	WindowsEventLogReader
HBase_2_RecordLookupService	XMLReader
Hive3ConnectionPool	XMLRecordSetWriter
HiveCatalogService	YamlTreeReader
ImpalaConnectionPool	ZendeskRecordSink
IPFIXReader	

Related Information

Supported NiFi processors
Supported NiFi reporting tasks
Supported NiFi parameter providers
Supported NiFi flow analysis rules
Supported NiFi flow registry clients

Supported NiFi reporting tasks

Review the list of Apache NiFi reporting tasks supported in Cloudera Flow Management 4.10.0.

Cloudera Flow Management is based on Apache NiFi and includes a set of reporting tasks, most of which are supported by Cloudera. To ensure optimal performance and reliable support, it is crucial to use only supported reporting tasks and avoid deploying unsupported ones in production environments.

- ControllerStatusReportingTask
- MonitorDiskUsage
- MonitorMemory
- PrometheusReportingTask
- QueryNiFiReportingTask
- ReportLineageToAtlas
- ScriptedReportingTask
- SiteToSiteBulletinReportingTask
- SiteToSiteMetricsReportingTask
- SiteToSiteProvenanceReportingTask
- SiteToSiteStatusReportingTask

Additional reporting tasks are developed and tested by the community but are not officially supported by Cloudera. Reporting tasks may be excluded for various reasons, including insufficient reliability, incomplete test coverage, community declaration of non-production readiness, or deviations from Cloudera best practices.

Related Information

Supported NiFi processors
Supported NiFi controller services
Supported NiFi parameter providers
Supported NiFi flow analysis rules
Supported NiFi flow registry clients

Supported NiFi parameter providers

Review the list of Apache NiFi parameter providers supported in Cloudera Flow Management 4.10.0.

Cloudera Flow Management is shipped with Apache NiFi and includes a set of parameter providers, most of which are supported by Cloudera. To ensure optimal performance and reliable support, it is crucial to use only supported parameter providers and avoid deploying unsupported ones in production environments.

- AwsSecretsManagerParameterProvider
- AzureKeyVaultSecretsParameterProvider
- CyberArkConjurParameterProvider
- DatabaseParameterProvider
- EnvironmentVariableParameterProvider
- FileParameterProvider
- GcpSecretManagerParameterProvider
- HashiCorpVaultParameterProvider

- OnePasswordParameterProvider
- PropertiesFileParameterProvider

Additional parameter providers are developed and tested by the community but are not officially supported by Cloudera. Parameter providers may be excluded for various reasons, including insufficient reliability, incomplete test coverage, community declaration of non-production readiness, or deviations from Cloudera best practices.

Related Information

Supported NiFi processors
Supported NiFi controller services
Supported NiFi reporting tasks
Supported NiFi flow analysis rules
Supported NiFi flow registry clients

Supported NiFi flow analysis rules

Review the list of Apache NiFi processors supported in Cloudera Flow Management 4.10.0.

Flow Analysis Rules allow analyzing components or parts of a flow to help maintain optimal design. Rules can be set as Recommendations (informational only) or Policies (enforceable). Recommendation violations are logged but do not affect functionality, while Policy violations invalidate components until resolved.

- DisallowComponentType
- DisallowConsecutiveConnectionsWithRoundRobinLB
- DisallowDeadEnd
- DisallowDeprecatedProcessor
- DisallowExtractTextForFullContent
- RecommendRecordProcessor
- RequireHandleHttpResponseAfterHandleHttpRequest
- RequireMergeBeforePutIceberg
- RestrictBackpressureSettings
- RestrictComponentNaming
- RestrictConcurrentTasksVsThreadPoolSizeInProcessors
- RestrictFlowFileExpiration
- RestrictProcessorConcurrency
- RestrictSchedulingForListProcessors
- RestrictThreadPoolSize
- RestrictYieldDurationForConsumeKafkaProcessors

Related Information

Supported NiFi processors
Supported NiFi controller services
Supported NiFi reporting tasks
Supported NiFi parameter providers
Supported NiFi flow registry clients

Supported NiFi flow registry clients

Review the list of Apache NiFi flow registry clients supported in Cloudera Flow Management 4.10.0.

NiFi Flow Registry clients are components in Apache NiFi that allow it to connect to external Flow Registries, services used to store and manage versioned dataflows.

- ClouderaDataFlowRegistryClient

- ClouderaFlowLibraryFlowRegistryClient
- GitHubFlowRegistryClient
- GitLabFlowRegistryClient
- NifiRegistryFlowRegistryClient

Related Information

Supported NiFi processors
Supported NiFi controller services
Supported NiFi reporting tasks
Supported NiFi parameter providers
Supported NiFi flow analysis rules

Supported Cloudera exclusive components

Review the list of Cloudera exclusive components supported in Cloudera Flow Management 4.10.0.

Cloudera Flow Management provides a set of NiFi components available only to Cloudera customers. These components provide additional functionality and are tailored to enhance the Cloudera NiFi experience. The list of these components is provided below.

Processors

- CaptureChangeDebeziumDB2
- CaptureChangeDebeziumMongoDB
- CaptureChangeDebeziumMySQL
- CaptureChangeDebeziumOracle
- CaptureChangeDebeziumPostgreSQL
- CaptureChangeDebeziumSQLServer
- ConsumeKafka2CDP
- ConsumeKafka2RecordCDP
- ConsumePLC
- ConvertProtobuf
- DeleteCDPObjectStore
- FetchCDPObjectStore
- FetchPLC
- GetJiraIssue
- InvokeGRPC
- ListCDPObjectStore
- ListenGRPC
- ListenNetFlow
- PromptAzureOpenAI
- PromptBedrock
- PromptClaude
- PromptOpenAI
- PublishKafka2CDP
- PublishKafka2RecordCDP
- PutCDPObjectStore
- PutClouderaHiveQL
- PutClouderaHiveStreaming
- PutClouderaORC
- PutIcebergCDC
- PutJiraIssue

- PutPLC
- SawmillTransformJSON
- SawmillTransformRecord
- SelectClouderaHiveQL
- TokenCount
- TriggerClouderaHiveMetaStoreEvent
- UpdateClouderaHiveTable
- UpdateDeltaLakeTable

Controller services

- ActiveMQJMSConnectionFactoryProvider
- ADLSIDBrokerCloudCredentialsProviderControllerService
- AWSIDBrokerCloudCredentialsProviderControllerService
- AzureBlobIDBrokerCloudCredentialsProviderControllerService
- AzureServiceBusJMSConnectionFactoryProvider
- CdpCredentialsProviderControllerService
- CdpOauth2AccessTokenProviderControllerService
- CiscoEmblemSyslogMessageReader
- ClouderaAttributeSchemaReferenceReader
- ClouderaAttributeSchemaReferenceWriter
- ClouderaEncodedSchemaReferenceReader
- ClouderaEncodedSchemaReferenceWriter
- ClouderaHiveConnectionPool
- ClouderaSchemaRegistry
- CMLLookupService
- EBCDICRecordReader
- GenericPLC4XConnectionPool
- ImpalaConnectionPool
- IPFIXReader
- JiraRecordSink
- PhoenixThickConnectionPool
- PhoenixThinConnectionPool
- PostgreSQLConnectionPool
- ProxyPLC4XConnectionPool
- RabbitMQJMSConnectionFactoryProvider
- RedshiftConnectionPool
- RESTCatalogService
- StandardPLC4XConnectionPool

Parameter providers

- CyberArkConjurParameterProvider
- PropertiesFileParameterProvider

Flow registry clients

- ClouderaDataFlowRegistryClient
- ClouderaFlowLibraryFlowRegistryClient

Download locations

You can download the Cloudera Flow Management software artifacts from the Cloudera Archive. There are different artifacts for different operating systems, standalone components, and Windows files.

Use the following tables to identify the Cloudera Flow Management 4.10.0 repository location for your operating system and operational objectives.

Note:

You must have credentials to download Cloudera Flow Management files. Your download credential is not the same as the credential you use to access the Cloudera Support Portal.

You can get download credentials in the following ways:

- Contact your Cloudera sales representative.
- Check the Welcome email you have received for your Cloudera Flow Management account.
- File a non-technical case on the Cloudera Support Portal for the Cloudera Support team to assist you.

Table 1: RHEL 8

File	Location
Manifest	https://archive.cloudera.com/p/cfm2/4.10.0.0/redhat8/yum/tars/parcel/manifest.json
Parcel	https://archive.cloudera.com/p/cfm2/4.10.0.0/redhat8/yum/tars/parcel/CFM-4.10.0.0-154-el8.parcel
Parcel sha file	https://archive.cloudera.com/p/cfm2/4.10.0.0/redhat8/yum/tars/parcel/CFM-4.10.0.0-154-el8.parcel.sha
CSD	**NiFi:** https://archive.cloudera.com/p/cfm2/4.10.0.0/redhat8/yum/tars/parcel/NIFI-2.3.0.4.10.0.0-154.jar **NiFi Registry:** https://archive.cloudera.com/p/cfm2/4.10.0.0/redhat8/yum/tars/parcel/NIFIREGISTRY-2.3.0.4.10.0.0-154.jar

Table 2: RHEL 9

File	Location
Manifest	https://archive.cloudera.com/p/cfm2/4.10.0.0/redhat9/yum/tars/parcel/manifest.json
Parcel	https://archive.cloudera.com/p/cfm2/4.10.0.0/redhat9/yum/tars/parcel/CFM-4.10.0.0-154-el9.parcel
Parcel sha file	https://archive.cloudera.com/p/cfm2/4.10.0.0/redhat9/yum/tars/parcel/CFM-4.10.0.0-154-el9.parcel.sha
CSD	**NiFi:** https://archive.cloudera.com/p/cfm2/4.10.0.0/redhat9/yum/tars/parcel/NIFI-2.3.0.4.10.0.0-154.jar **NiFi Registry:** https://archive.cloudera.com/p/cfm2/4.10.0.0/redhat9/yum/tars/parcel/NIFIREGISTRY-2.3.0.4.10.0.0-154.jar

Table 3: Ubuntu 20

File	Location
Manifest	https://archive.cloudera.com/p/cfm2/4.10.0.0/ubuntu20/apt/tars/parcel/manifest.json
Parcel	https://archive.cloudera.com/p/cfm2/4.10.0.0/ubuntu20/apt/tars/parcel/CFM-4.10.0.0-154-focal.parcel
Parcel SHA file	https://archive.cloudera.com/p/cfm2/4.10.0.0/ubuntu20/apt/tars/parcel/CFM-4.10.0.0-154-focal.parcel.sha

File	Location
CSD	**NiFi:** https://archive.cloudera.com/p/cfm2/4.10.0.0/ubuntu20/apt/tars/parcel/NIFI-2.3.0.4.10.0.0-154.jar **NiFi Registry:** https://archive.cloudera.com/p/cfm2/4.10.0.0/ubuntu20/apt/tars/parcel/NIFIREGISTRY-2.3.0.4.10.0.0-154.jar

Table 4: Ubuntu 22

File	Location
Manifest	https://archive.cloudera.com/p/cfm2/4.10.0.0/ubuntu22/apt/tars/parcel/manifest.json
Parcel	https://archive.cloudera.com/p/cfm2/4.10.0.0/ubuntu22/apt/tars/parcel/CFM-4.10.0.0-154-jammy.parcel
Parcel SHA file	https://archive.cloudera.com/p/cfm2/4.10.0.0/ubuntu22/apt/tars/parcel/CFM-4.10.0.0-154-jammy.parcel.sha
CSD	**NiFi:** https://archive.cloudera.com/p/cfm2/4.10.0.0/ubuntu22/apt/tars/parcel/NIFI-2.3.0.4.10.0.0-154.jar **NiFi Registry:** https://archive.cloudera.com/p/cfm2/4.10.0.0/ubuntu22/apt/tars/parcel/NIFIREGISTRY-2.3.0.4.10.0.0-154.jar

Table 5: Standalone components (OS agnostic)

File	Location
NiFi (.tar.gz)	https://archive.cloudera.com/p/cfm2/4.10.0.0/redhat8/yum/tars/cdf_extensions/nifi-2.3.0.4.10.0.0-154-bin.tar.gz
NiFi (.tar.gz.sha256)	https://archive.cloudera.com/p/cfm2/4.10.0.0/redhat8/yum/tars/cdf_extensions/nifi-2.3.0.4.10.0.0-154-bin.tar.gz.sha256
NiFi Registry (.tar.gz)	https://archive.cloudera.com/p/cfm2/4.10.0.0/redhat8/yum/tars/cdf_extensions/nifi-registry-2.3.0.4.10.0.0-154-bin.tar.gz
NiFi Registry (.zip)	https://archive.cloudera.com/p/cfm2/4.10.0.0/redhat8/yum/tars/cdf_extensions/nifi-registry-2.3.0.4.10.0.0-154-bin.zip

Table 6: Windows files

File	Location
NiFi MSI	https://archive.cloudera.com/p/cfm2/4.10.0.0/windows/nifi-4.10.0.0-154.msi
NiFi MSI SHA	https://archive.cloudera.com/p/cfm2/4.10.0.0/windows/nifi-4.10.0.0-154.msi.sha256

Unsupported features

Review the list of features that are not supported in Cloudera Flow Management 4.10.0.

The following features are developed and tested by the Cloudera community but are not officially supported by Cloudera. These features are excluded for a variety of reasons, including insufficient reliability or incomplete test case coverage, declaration of non-production readiness by the community at large, and feature deviation from Cloudera best practices. Do not use these features in your production environments.

Unsupported NiFi components

NARs

NiFi 1 custom NARs cannot be successfully loaded into NiFi 2. If your NiFi setup includes custom NARs, it is a requirement to update your dependencies to align with NiFi 2. This entails making the necessary adjustments and rebuilding your NARs using Java 21. The below components are not supported and should not be used anymore.

- nifi-cybersecurity-nar
- nifi-email-nar
- nifi-hive-nar
- nifi-rethinkdb-nar
- nifi-influxdb-nar
- nifi-ccda-nar
- nifi-html-nar
- nifi-ignite-nar
- nifi-tcp-nar
- nifi-riemann-nar
- nifi-spring-nar
- nifi-kite-nar
- nifi-rules-action-handler-nar
- nifi-azure-nar
- nifi-easyrules-nar
- nifi-metrics-reporting-nar
- nifi-other-graph-services-nar
- nifi-hbase_1_1_2-client-service-nar
- nifi-scripting-nar
- nifi-ambari-nar
- nifi-sql-reporting-nar
- nifi-aws-nar
- nifi-accumulo-nar
- nifi-solr-nar
- nifi-accumulo-service-nar
- nifi-datadog-nar
- nifi-atlas-nar
- nifi-beats-nar
- nifi-standard-nar
- nifi-language-translation-nar
- nifi-livy-nar
- nifi-pulsar-nar

Rules engine processors

Rules engine components and handlers are removed in NiFi 2, so the below processors are not supported and should not be used anymore.

- ActionHandlerLookup
- AlertHandler
- EasyRulesEngineProvider
- EasyRulesEngineService
- ExpressionHandler
- LogHandler
- RecordSinkHandler
- ScriptedActionHandler
- ScriptedRulesEngine

Unsupported customizations

Cloudera cannot guarantee that default NiFi processors are compatible with proprietary protocol implementations or proprietary interface extensions. For example, Cloudera supports interfaces like JMS and JDBC that are built around standards, specifications, or open protocols, but does not support customizations of those interfaces, or proprietary extensions built on top of those interfaces.

Behavioral changes

Learn about behavioral changes in Cloudera Flow Management 4.10.0.

NiFi 2.0 introduces a lot of significant changes and enhancements, including some breaking changes. It is important to familiarize yourself with the following points before migrating your existing flows.

Java 21

Java 21 is the minimum Java version required with NiFi 2.0, so make sure you have Java 21 installed on your NiFi nodes before upgrading.

Templates and XML flow definitions

The concept of templates in NiFi has been deprecated, and the XML templates are stored in memory in NiFi as well as in the persisted flow definition.

Additionally, flow.xml.gz no longer exists, only flow.json.gz can be used in NiFi clusters for defining flows in the canvas.

If you have templates, export those templates as JSON definitions or version the templates into a NiFi Registry instance. The best practice is to use a NiFi Registry in combination with NiFi when it comes to version control and share / reuse flow definitions.

Custom components / NARs

Although not certain, it is very likely that a custom NAR designed for NiFi 1 will not be successfully loaded into NiFi 2. If your NiFi setup includes custom components or NARs, it is a requirement to update your dependencies to align with NiFi 2. This entails making the necessary adjustments and rebuilding your NARs using Java 21.

Variables replaced by parameters

Variables and the variable registry have been removed from NiFi. Only Parameter contexts and parameters are available for use going forward. In future Cloudera Flow Management releases, tooling will be provided to help with the conversion of variables to parameters. In the meantime, this conversion should be done manually when migrating flows to NiFi 2. Any variables left will simply be ignored when loading the flow definition.

Event driven thread pool no longer exists

The event driven scheduling strategy was an option available on some processors. This was an experimental feature in NiFi and did not prove to bring any significant performance improvements. The event driven thread pool has been removed, leaving only the time driven thread pool available. Any components previously configured using the event driven scheduling strategy should be switched to the time driven scheduling strategy.

Removed languages in scripted components

In NiFi 2.0, support for certain languages in scripted components has been removed. The affected languages are: ECMAScript, Lua, Ruby, and Python. It is recommended to switch to Groovy or to leverage the new Python API feature for developing processors.

Removed components and replacement options

The following list contains the list of the components that have been removed between clusters based on NiFi 1.26 and clusters based on NiFi 2.0, along with the recommended alternatives where available.

- Processors
  - Base64EncodeContent => EncodeContent
  - CompareFuzzyHash => no replacement
  - ConsumeEWS => no replacement
  - ConsumeKafka_1_0 => ConsumeKafka_2_6
  - ConsumeKafka_2_0 => ConsumeKafka_2_6
  - ConsumeKafkaRecord_1_0 => ConsumeKafkaRecord_2_6
  - ConsumeKafkaRecord_2_0 => ConsumeKafkaRecord_2_6
  - ConvertAvroSchema => ConvertRecord
  - ConvertAvroToORC => no replacement
  - ConvertCSVToAvro => ConvertRecord
  - ConvertExcelToCSVProcessor => ConvertRecord with ExcelReader
  - ConvertJSONToAvro => ConvertRecord
  - CryptographicHashAttribute => UpdateAttribute
  - DeleteAzureBlobStorage => DeleteAzureBlobStorage_v12
  - DeleteRethinkDB => no replacement
  - EncryptContent => EncryptContentAge or EncryptContentPGP
  - ExecuteInfluxDBQuery => use Influx Data NARs for NiFi
  - ExtractCCDAAttributes => no replacement
  - FetchAzureBlobStorage => FetchAzureBlobStorage_v12
  - FetchElasticsearchHttp => GetElasticsearch
  - FuzzyHashContent => no replacement
  - GetAzureQueueStorage => GetAzureQueueStorage_v12
  - GetHTMLElement => no replacement
  - GetHTTP => InvokeHTTP
  - GetIgniteCache => no replacement
  - GetJMSQueue => ConsumeJMS
  - GetJMSTopic => ConsumeJMS
  - GetRethinkDB => no replacement
  - GetTCP => no replacement
  - GetTwitter => ConsumeTwitter
  - HashAttribute => CryptographicHashAttribute
  - HashContent => CryptographicHashContent
  - InferAvroSchema => ExtractRecordSchema
  - ListAzureBlobStorage => ListAzureBlobStorage_v12
  - ModifyHTMLElement => no replacement
  - PostHTTP => InvokeHTTP
  - PostSlack => PublishSlack
  - PublishKafka_1_0 => PublishKafka_2_6
  - PublishKafka_2_0 => PublishKafka_2_6
  - PublishKafkaRecord_1_0 => PublishKafkaRecord_2_6
  - PublishKafkaRecord_2_0 => PublishKafkaRecord_2_6
  - PutAzureBlobStorage => PutAzureBlobStorage_v12
  - PutAzureQueueStorage => PutAzureQueueStorage_v12
  - PutBigQueryBatch => PutBigQuery
  - PutBigQueryStreaming => PutBigQuery
  - PutElasticsearchHttp => PutElasticsearchJson
  - PutElasticsearchHttpRecord => PutElasticsearchRecord
  - PutHiveQL => PutClouderaHiveQL
  - PutHiveStreaming => PutClouderaHiveStreaming

    - PutHTMLElement => no replacement
    - PutIgniteCache => no replacement
    - PutInfluxDB => use Influx Data NARs for NiFi
    - PutJMS => PublishJMS
    - PutRethinkDB => no replacement
    - PutRiemann => no replacement
    - PutSlack => PublishSlack
    - QueryElasticsearchHttp => PaginatedJsonQueryElasticsearch
    - ScrollElasticsearchHttp => SearchElasticsearch
    - SelectHiveQL => SelectClouderaHiveQL
    - SpringContextProcessor => no replacement
    - StoreInKiteDataset => no replacement
    - UpdateHiveTable => UpdateClouderaHiveTable
- Controller services
    - ActionHandlerLookup => no replacement
    - AlertHandler => no replacement
    - AzureStorageCredentialsControllerService => AzureStorageCredentialsControllerService_v12
    - AzureStorageCredentialsControllerServiceLookup => AzureStorageCredentialsControllerServiceLookup_v12
    - AzureStorageEmulatorCredentialsControllerService => no replacement
    - EasyRulesEngineProvider => no replacement
    - EasyRulesEngineService => no replacement
    - ExpressionHandler => no replacement
    - GraphiteMetricReporterService => no replacement
    - GremlinClientService => no replacement
    - HBase_1_1_2_ClientMapCacheService => HBase_2_ClientMapCacheService
    - HBase_1_1_2_ClientService => HBase_2_ClientService
    - HBase_1_1_2_ListLookupService => no replacement
    - HBase_1_1_2_RecordLookupService => HBase_2_RecordLookupService
    - HiveConnectionPool => ClouderaHiveConnectionPool
    - HortonworksSchemaRegistry => ClouderaSchemaRegistry
    - KafkaRecordSink_1_0 => KafkaRecordSink_2_6
    - KafkaRecordSink_2_0 => KafkaRecordSink_2_6
    - KeytabCredentialsService => KerberosKeytabUserService
    - LogHandler => no replacement
    - OAuth2TokenProviderImpl => StandardOauth2AccessTokenProvider
    - OpenCypherClientService => no replacement
    - RecordSinkHandler => no replacement
    - ScriptedActionHandler => no replacement
    - ScriptedRulesEngine => no replacement
- Reporting tasks
    - AmbariReportingTask => no replacement
    - MetricsEventReportingTask => no replacement
    - MetricsReportingTask => no replacement

- Components with new coordinates
  - InvokeGRPC => moved into nifi-cdf-grpc-nar
  - ListenGRPC => moved into nifi-cdf-grpc-nar
  - KerberosKeytabUserService => moved into nifi-kerberos-user-service-nar
  - KerberosPasswordUserService => moved into nifi-kerberos-user-service-nar
  - KerberosTicketCacheUserService => moved into nifi-kerberos-user-service-nar

  Tooling will be provided in upcoming Cloudera Flow Management releases to automatically handle these changes. Currently, two options are available:

  - Manually edit the flow.json.gz file to update the coordinates of the impacted components.
  - Make the changes after the flow is imported in NiFi 2.0 by replacing the ghost components with the new implementations for each instance of the components listed above.
- Pulsar components

  All Pulsar components have been removed. You can download the NARs from a public Maven repository and deploy them as custom NARs.

  - nifi-pulsar-nar
  - nifi-pulsar-client-service-nar

Breaking changes in NiFi 2

Learn about the breaking changes Apache NiFi 2 will bring to Cloudera Flow Management.

Flow design level breaking changes

Moved components

In NiFi 2, some components have been relocated within the Apache NiFi repository, resulting in changes to the bundle coordinates for the associated NAR files.

Unfortunately, no pre-upgrade actions can fully prevent these breaking changes. You will need to update the flow.json.gz file with new coordinates. Cloudera's upcoming NiFi Migration Tooling is designed to automate as many changes as possible to help the upgrade process. Some changes will still require manual handling, so it is highly recommended to run the pre-upgrade check script to identify any potential issues or impacted components before proceeding with the upgrade.

JoltTransformJSON processor

For Previous coordinates

```
…
            "type": "org.apache.nifi.processors.standard.JoltT
ransformJSON",
            "bundle": {
              "group": "org.apache.nifi",
              "artifact": "nifi-standard-nar",
              "version": "1.27.0"
            }
…
```

For New coordinates

```
…
            "type": "org.apache.nifi.processors.jolt.JoltTrans
formJSON",
            "bundle": {
              "group": "org.apache.nifi",
              "artifact": "nifi-jolt-nar",
```

```
              "version": "2.0.0"
            }
…
```

JoltTransformRecord processor

For Previous coordinates

```
…
            "type": "org.apache.nifi.processors.jolt.record.Jo
ltTransformRecord",
            "bundle": {
              "group": "org.apache.nifi",
              "artifact": "nifi-jolt-record-nar",
              "version": "1.27.0"
            }
…
```

For New coordinates

```
…
            "type": "org.apache.nifi.processors.jolt.JoltTrans
formRecord",
            "bundle": {
              "group": "org.apache.nifi",
              "artifact": "nifi-jolt-nar",
              "version": "2.0.0"
            }
…
```

FileParameterProvider renamed to KubernetesSecretParameterProvider

For Previous coordinates

```
…
            "type": "org.apache.nifi.parameter.FileParameterPr
ovider",
            "bundle": {
              "group": "org.apache.nifi",
              "artifact": "nifi-standard-nar",
              "version": "1.27.0"
            }
…
```

For New coordinates

```
…
            "type": "org.apache.nifi.parameter.KubernetesSecre
tParameterProvider",
            "bundle": {
              "group": "org.apache.nifi",
              "artifact": "nifi-standard-nar",
              "version": "2.0.0"
            }
…
```

Removed key components

- Kafka processors:

  All Kafka processors in Apache NiFi have been removed and got replaced by new components using a controller service-based approach. This change is a significant breaking change, as it does not allow for a non-breaking upgrade. To ease this transition, Cloudera has preserved the Kafka 2.6 processors without altering the bundle coordinates. However, adjustments to the Kerberos configuration will still be necessary (see the details below). This approach provides time to transition to the new Kafka components while on NiFi 2.
- Hive components:

  All Hive-related components in Apache NiFi have been removed. Cloudera has introduced specific components downstream to align with the Hive version distributed as part of CDP. You will need to perform proper bundle coordinate updates in the flow.json.gz file to migrate to these new components. Both the old and new components are available in the latest NiFi 1.x releases, so you are advised to switch to the new components while still using NiFi 1.x.

Components with Kerberos configuration changes

Kerberos authentication in NiFi requires presenting a Kerberos credential which can be in one of the following forms:

- Principal + Keytab: The keytab, stored on disk, contains the client's secret key. This credential is used by the application to authenticate and obtain the Ticket Granting Ticket (TGT).
- Principal + Password: The client's secret key is derived from a password, so no keytab is stored on disk. Otherwise, this method functions similarly to the keytab-based approach.
- Principal + Ticket cache: The TGT must be acquired externally and stored in a ticket cache, which the application uses. The application itself is unaware of the keytab or password and is not responsible for handling TGT acquisition.

Historically, NiFi supported the following Kerberos configuration options:

- Kerberos Principal + Kerberos Keytab: Component-level properties supporting keytab-based credential type
- Kerberos Credentials Service: A property referencing the KerberosCredentialsService controller service interface with the following implementation: KeytabCredentialsService for keytab-based credential type
- Kerberos Principal + Kerberos Password: Component-level properties supporting password-based credentials
- Kerberos User Service: A property referencing the KerberosUserService controller service interface, with implementations for all credential types:
  - KerberosKeytabUserService for keytab-based credentials
  - KerberosPasswordUserService for password-based credentials
  - KerberosTicketCacheUserService for ticket cache-based credentials.

In NiFi 2, only the Kerberos User Service is retained because it can accommodate all credential types (keytab, password, ticket cache) with a single property on the component. The other configuration options have been removed in NiFi 2. For more information, see NIFI-13510.

Affected components in Cloudera Flow Management 2.1.7:

The tables below list the affected components and their legacy Kerberos properties, also indicating whether these components are available in Cloudera Flow Management 4.x.

Table 7: Cloudera Object Store processors

Component Type	Compo
org.apache.nifi.processors.hadoop.ListCDPObjectStore	nifi-cdf-
org.apache.nifi.processors.hadoop.FetchCDPObjectStore	nifi-cdf-

Component Type	Compo
org.apache.nifi.processors.hadoop.PutCDPObjectStore	nifi-cdf-
org.apache.nifi.processors.hadoop.DeleteCDPObjectStore	nifi-cdf-

Table 8: Hadoop processors

Component Type	Compo
org.apache.nifi.processors.hadoop.ListHDFS	nifi-hado
org.apache.nifi.processors.hadoop.FetchHDFS	nifi-hado
org.apache.nifi.processors.hadoop.PutHDFS	nifi-hado
org.apache.nifi.processors.hadoop.DeleteHDFS	nifi-hado
org.apache.nifi.processors.hadoop.GetHDFS	nifi-hado
org.apache.nifi.processors.hadoop.MoveHDFS	nifi-hado
org.apache.nifi.processors.hadoop.inotify.GetHDFSEvents	nifi-hado
org.apache.nifi.processors.hadoop.GetHDFSFileInfo	nifi-hado
org.apache.nifi.processors.hadoop.GetHDFSSequenceFile	nifi-hado
org.apache.nifi.processors.hadoop.CreateHadoopSequenceFile	nifi-hado
org.apache.nifi.processors.parquet.FetchParquet	nifi-parq
org.apache.nifi.processors.parquet.PutParquet	nifi-parq
org.apache.nifi.processors.orc.PutORC	nifi-hive

Table 9: HBase services

Component Type	Compo
org.apache.nifi.hbase.HBase_1_1_2_ClientService	nifi-hbas
org.apache.nifi.hbase.HBase_2_ClientService	nifi-hbas

Table 10: Hive components

Component Type	Compo
org.apache.nifi.dbcp.hive.HiveConnectionPool	nifi-hive
org.apache.nifi.dbcp.hive.Hive3ConnectionPool	nifi-hive
org.apache.nifi.processors.hive.PutHiveStreaming	nifi-hive
org.apache.nifi.processors.hive.PutHive3Streaming	nifi-hive

Table 11: Kafka processors

Component Type	Compo
org.apache.nifi.processors.kafka.pubsub.ConsumeKafka_1_0	nifi-kafk
org.apache.nifi.processors.kafka.pubsub.PublishKafka_1_0	nifi-kafk
org.apache.nifi.processors.kafka.pubsub.ConsumeKafkaRecord_1_0	nifi-kafk
org.apache.nifi.processors.kafka.pubsub.PublishKafkaRecord_1_0	nifi-kafk

Component Type	Compo
org.apache.nifi.processors.kafka.pubsub.ConsumeKafka_2_0	nifi-kafk
org.apache.nifi.processors.kafka.pubsub.PublishKafka_2_0	nifi-kafk
org.apache.nifi.processors.kafka.pubsub.ConsumeKafkaRecord_2_0	nifi-kafk
org.apache.nifi.processors.kafka.pubsub.PublishKafkaRecord_2_0	nifi-kafk
org.apache.nifi.processors.kafka.pubsub.ConsumeKafka_2_6	nifi-kafk
org.apache.nifi.processors.kafka.pubsub.PublishKafka_2_6	nifi-kafk
org.apache.nifi.processors.kafka.pubsub.ConsumeKafkaRecord_2_6	nifi-kafk
org.apache.nifi.processors.kafka.pubsub.PublishKafkaRecord_2_6	nifi-kafk
org.apache.nifi.processors.kafka.pubsub.ConsumeKafka2CDP	nifi-cdf-
org.apache.nifi.processors.kafka.pubsub.PublishKafka2CDP	nifi-cdf-
org.apache.nifi.processors.kafka.pubsub.ConsumeKafkaRecord2CDP	nifi-cdf-
org.apache.nifi.processors.kafka.pubsub.PublishKafkaRecord2CDP	nifi-cdf-

Table 12: Solr processors

Component Type	Compo
org.apache.nifi.processors.solr.GetSolr	nifi-solr-
org.apache.nifi.processors.solr.QuerySolr	nifi-solr-
org.apache.nifi.processors.solr.PutSolrContentStream	nifi-solr-
org.apache.nifi.processors.solr.PutSolrRecord	nifi-solr-

Table 13: Other processors

Component Type	Compo
org.apache.nifi.accumulo.controllerservices.AccumuloService	nifi-accu
org.apache.nifi.dbcp.DBCPConnectionPool	nifi-dbcp
org.apache.nifi.dbcp.HadoopDBCPConnectionPool	nifi-hado
org.apache.nifi.schemaregistry.hortonworks.HortonworksSchemaRegistry	nifi-hwx
org.apache.nifi.controller.kudu.KuduLookupService	nifi-kudu
org.apache.nifi.controller.livy.LivySessionController	nifi-livy-
org.apache.nifi.processors.kudu.PutKudu	nifi-kudu
org.apache.nifi.atlas.reporting.ReportLineageToAtlas	nifi-atlas

Flow migration for Kerberos configuration changes

To ensure compatibility with NiFi 2, you will need to migrate your flow by creating a new Kerberos UserService controller service based on the old controller service or component level properties. This process involves transitioning from legacy component-level properties to the updated service and clearing outdated configurations.

Migration steps for **Kerberos Principal + Kerberos Keytab**:

1. Create service: Set up a KerberosKeytabUserService using the existing Kerberos Principal and Kerberos Keytab component-level properties.
2. Update reference: Link the new service to the component's Kerberos User Service property.

3. Remove legacy properties: Clear the old property values from Kerberos Principal and Kerberos Keytab.

Migration steps for **Kerberos Credentials Service**:

1. Create service: Set up a KerberosKeytabUserService based on the properties from the KeytabCredentialsService.
2. Update reference: Point the component's Kerberos User Service property to the new service.
3. Remove legacy service: Clear the old Kerberos Credentials Service reference property and delete the outdated service.

Migration steps for **Kerberos Principal + Kerberos Password**:

1. Create service: Set up a KerberosPasswordUserService using the Kerberos Principal and Kerberos Password component-level properties.
2. Update reference: Link the new service to the component’s Kerberos User Service property.
3. Remove legacy properties: Clear the old property values from Kerberos Principal and Kerberos Password.

Components requiring initial code-level changes

For certain components, the Kerberos User Service property is not yet available. These components will require an initial code-level update before migration.

- Kafka_1_0 processors
- Kafka_2_0 processors
- CDPObjectStore processors
- ReportLineageToAtlas
- KuduLookupService
- LivySessionController

Migration to Cloudera-specific components

As part of the migration, some components will transition to new, Cloudera-specific types. The Kerberos property migration should be included in this transition.

- HortonworksSchemaRegistry => ClouderaSchemaRegistry
- Hive[3]ConnectionPool => ClouderaHiveConnectionPool
- PutHive[3]Streaming => PutClouderaHiveStreaming

Scripted components

In NiFi 2, support for certain languages in scripted components has been removed. The affected languages include ECMAScript, Lua, Ruby, and Python. Cloudera recommends to switch to Groovy or leverage the new Python API feature for developing processors. The following components are impacted by this change:

- org.apache.nifi.processors.script.ExecuteScript
- org.apache.nifi.processors.script.InvokeScriptedProcessor
- org.apache.nifi.processors.script.ScriptedFilterRecord
- org.apache.nifi.processors.script.ScriptedPartitionRecord
- org.apache.nifi.processors.script.ScriptedTransformRecord
- org.apache.nifi.processors.script.ScriptedValidateRecord
- org.apache.nifi.lookup.script.ScriptedLookupService
- org.apache.nifi.record.script.ScriptedReader
- org.apache.nifi.record.script.ScriptedRecordSetWriter
- org.apache.nifi.record.sink.script.ScriptedRecordSink
- org.apache.nifi.lookup.script.SimpleScriptedLookupService
- org.apache.nifi.reporting.script.ScriptedReportingTask

Important: As of now, the Python API in NiFi 2 is limited to creating processors. Writing controller services or reporting tasks using Python is not supported.

Custom components

If your NiFi environment includes custom components or NARs developed for NiFi 1.x, they are unlikely to be compatible with NiFi 2. To ensure compatibility, you must update your dependencies to align with NiFi 2 and rebuild your NARs using Java 21. This update is essential for a successful transition to the new version.

Flow controller level breaking changes

Transition from variables to parameters

Variables and the variable registry are removed in NiFi 2 due to their inherent limitations, such as requiring expression language support to reference a variable and the inability to store sensitive values. You can use parameter contexts instead, which have been significantly enhanced over recent years. For example, the addition of the Parameter Context Provider allows for sourcing parameter values from external stores (like HashiCorp Vault or cloud provider vaults).

This change is one of the most impactful in NiFi 2, which will require rework on existing data flows. However, it also presents an opportunity to optimize the organization of parameters, allowing you to split them into multiple parameter contexts and use inheritance when sharing parameters across different use cases.

Cloudera will provide automated tools within the NiFi Migration Tooling to assist with transitioning from variables to parameters.

Removal of XML templates

The concept of XML templates is being phased out in NiFi 2. Historically, these templates were stored in memory as well as in the flow definition files (flow.xml.gz and flow.json.gz). This caused significant issues for some NiFi users, especially those managing numerous large templates with thousands of components. Removing templates from NiFi will enhance stability and reduce memory usage.

If you use templates in your NiFi 1.x clusters, you should export your existing templates as JSON definitions or version them into a NiFi Registry instance to prepare for this change. Using NiFi Registry is the recommended best practice for version control, sharing, and reusing flow definitions.

If your template is a process group:

1. Drag and drop the template onto the canvas.
2. Right-click it, and choose to export it as a flow definition (JSON file) or start version control in your NiFi Registry, if you have one configured.

If your template is not a process group (just a flow with components):

1. Drag and drop a process group onto the canvas.
2. Go into that process group and drag and drop your template within that process group.
3. Go back to the parent process group containing your template and export it as a flow definition or start version control on it.

Cloudera will provide automated tools within the NiFi Migration Tooling to help manage the migration of templates.

Discontinuation of event driven thread pool

The Event Driven Scheduling Strategy, which was available for some processors in previous versions of NiFi, is being removed in NiFi 2. This feature was experimental and did not demonstrate significant performance improvements.

If you are using this scheduling strategy, you will need to update your components to use the time driven scheduling strategy instead. You can identify components using the Event Driven strategy by searching for “event” in the NiFi search bar.

NiFi framework breaking changes in NiFi 2

Java 21 compatibility

Java 21 is the minimum Java version required with NiFi 2. While most versions of NiFi 1.x may work with Java 21, it is not officially supported to run NiFi 1.x with Java 21 for extended periods in preparation for the upgrade. The transition to Java 21 should happen as part of the upgrade process to NiFi 2. You have to ensure that your environment is ready for Java 21 before initiating the upgrade.

Transition from flow.xml.gz to flow.json.gz

In NiFi 2, flow.xml.gz, which has been a cornerstone of flow configuration storage, is completely phased out and replaced by flow.json.gz. While the two files have coexisted in many NiFi 1.x releases, NiFi 2 will exclusively use JSON-based flow representations. Before upgrading, it is recommended to back up your flow.json.gz file. During the upgrade, only work with this JSON file if any updates are needed.

Repository encryption removal

The ability to encrypt repositories has been removed in NiFi 2.

NiFi Toolkit changes

The NiFi Toolkit will also undergo changes in NiFi 2, with some features being removed.

Dynamic parameter retrieval with parameter providers

In NiFi 2, the values associated with parameters retrieved by parameter providers are no longer stored within the flow.json.gz file. Instead, these values are retrieved on-demand or during NiFi's startup and are only stored in memory. This change enhances security and reduces the size of flow files, but it also means that parameter values will need to be available from their source each time NiFi is started or when the parameters are accessed. It is important to ensure that the external sources for these parameters are reliable and accessible to avoid disruptions during NiFi operations. For more information, see NIFI-13560.