DATA BROKER

Interface Control Document with Web Services

Texas Department of Public Safety

v1.0.0

October 5, 2021

Table of Contents

1. Scope

This document provides instructions for submission of National Data Exchange (N-DEx), National Incident-Based Reporting System (NIBRS), and Texas NIBRS information to the Data Broker provided by the Texas Department of Public Safety. The document details how to connect to the Data Broker and the steps for approval to submit N-DEx, NIBRS, and Texas NIBRS data.

The Data Broker provides for the full interactions between law enforcement submitters, Texas's Texas Data Exchange (TDEx) program, the Federal Bureau of Investigation's (FBI's) N-DEx Program, and Texas's Uniform Crime Reporting (UCR) program.

The process for submission of data to the Data Broker for inclusion in the FBI's N-DEx and/or UCR programs and the Texas NIBRS specifications includes:

1. Law enforcement Entities originate N-DEx records;
2. The Data Broker accepts N-DEx records and NIBRS records;
3. The FBI-CJIS Division's N-DEx Program accepts N-DEx records, and
4. The Texas UCR Repository accepts federal and Texas NIBRS records.

The Data Broker processes are uni-directional, flowing from law enforcement Entities to the Data Broker to the FBI's N-DEx and to the Texas UCR repository.

2. Data Broker

The Data Broker provides web services to submit N-DEx and NIBRS data.

Data Broker Web Services	Description
SubmitNDExDocument	This process accepts N-DEx data via a NIEM N-DEx 4.0 IEPD formatted document. It returns a successfully submitted status.
SubmitNIBRSDocument	This process accepts NIBRS data via a NIEM NIBRS 2019.2 IEPD formatted document. It returns a successfully submitted status.
SubmitCombinedDocument	This process accepts a file containing both N-DEx and NIBRS data via an N-DEx 4.0 IEPD formatted document. It returns a successfully submitted status.

N-DEx and NIBRS data is submitted to the Data Broker Services via compliant National Information Exchange Model (NIEM) Information Exchange Package Documentation (IEPD) documents.

2.1 Interface Overview

Texas Entities interface with the Data Broker web services via the public internet using the Data Broker API endpoints described below.

2.1.1 Availability

The Data Broker is available 24/7/365 days a year, excluding maintenance periods.

Contributing Entity contacts will be notified by DPS staff via email at least 72 hours in advance of scheduled maintenance. In the event that there needs to be unscheduled maintenance due to unforeseen circumstances, Contributing Entity contacts will be notified.

Contributing Entity contacts will be notified via the email addresses that are currently on file.

All transactions with the Data Broker are handled via HTTP REST protocol over the public internet. The messages are encrypted between the client and service.

2.1.2 Format

Data Broker report submissions must be structured to the NIEM IEPD markup standard. The NIEM IEPD documentation is available on the Data Broker website in the “Getting Started” section.

The state of Texas requires specific data elements to be included with certain types of NIBRS transactions. Those data elements conform to the IEPD standards. The rules governing those data elements and how they are incorporated into the NIBRS submission are described in the Texas Specific NIBRS Requirements document available on the Data Broker website in the “Getting Started” section.

2.1.3 Data Broker Accounts

2.1.3.1 Establishing a Data Broker Account

The following describes the steps to establish a Data Broker account:

1. The Contributing Entity requests access to the Data Broker via email sent to databroker@dps.texas.gov.
2. The DPS representative creates a process account for the Contributing Entity.
3. The Contributing Entity’s account receives an email with instructions to complete the Data Broker account setup.

2.1.3.2 Logging In

Once the process account is established, an associated user can log in to the Data Broker website at https://databrokeruat.dps.texas.gov during testing and certification, and at https://databroker.dps.texas.gov in production.

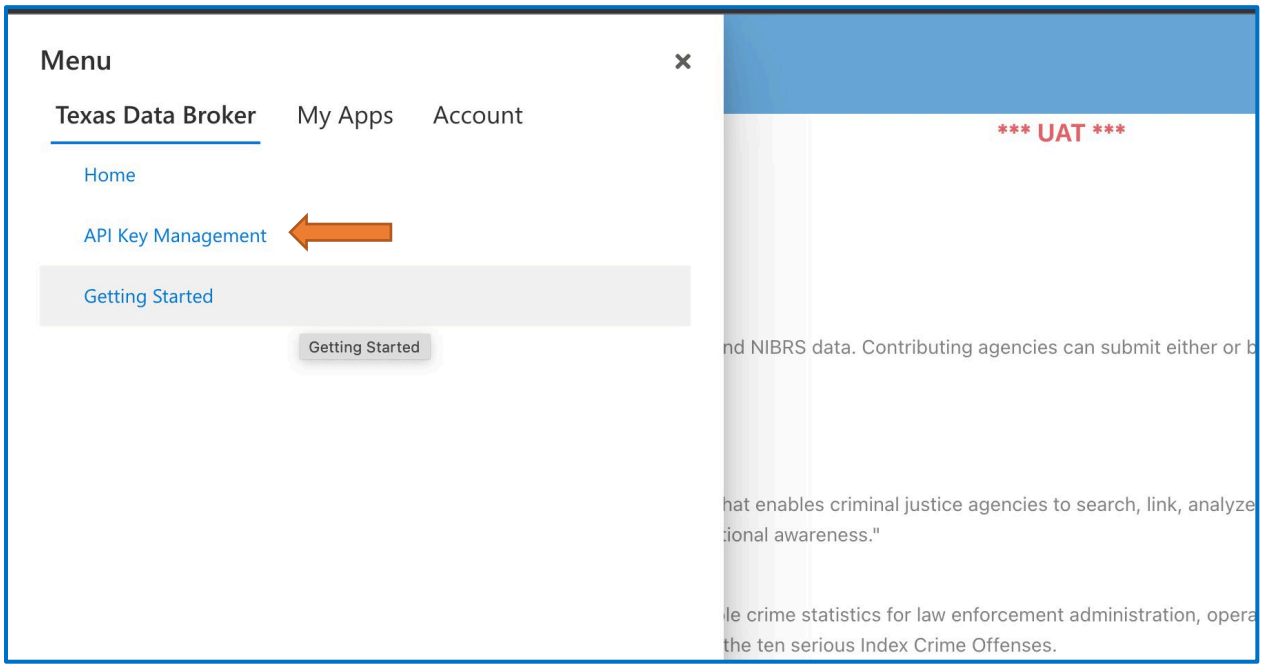
Upon login, the Texas Data Broker Getting Started page appears. This page has information about the N-DEx and NIBRS programs and links to download the IEPD packages and other documentation.

2.1.3.3 API Key Management

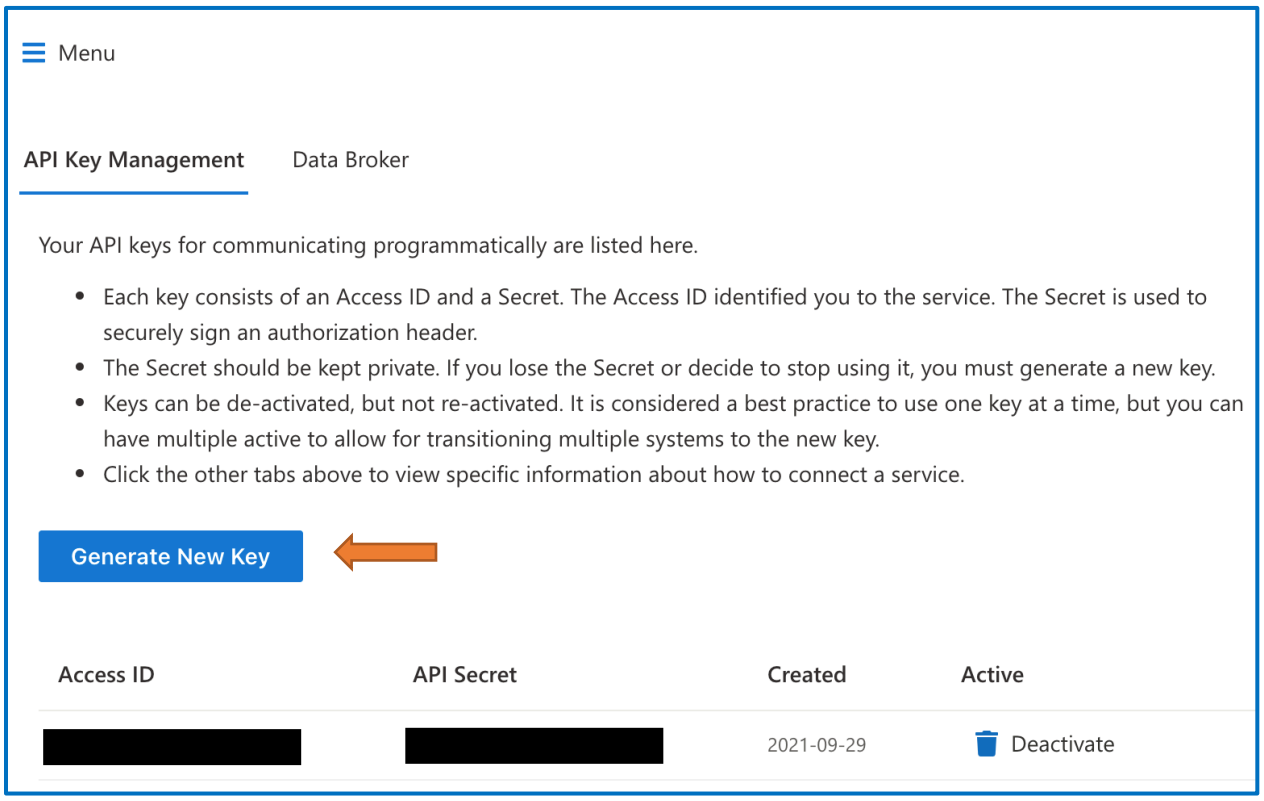
Uploads to the Data Broker use a shared secret process, which requires an Access ID and a private Secret. API Key Management is used to obtain the information and keys used to upload files to the application.

To access the API Key Management:

- From the Texas Data Broker Menu, choose **API Key Management.**

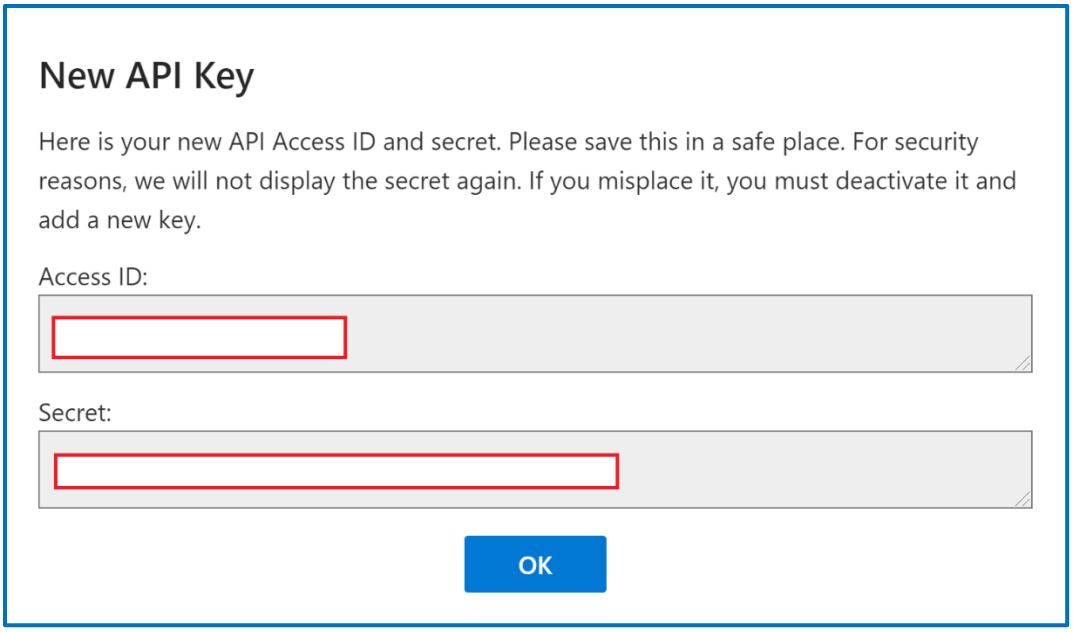


- From the initial screen, click on **Generate New Key** to get the keys necessary to submit a document.



- This page lists any existing keys you have generated. Clicking the **Deactivate** icon will disable a key, for example, if the key is lost or compromised.

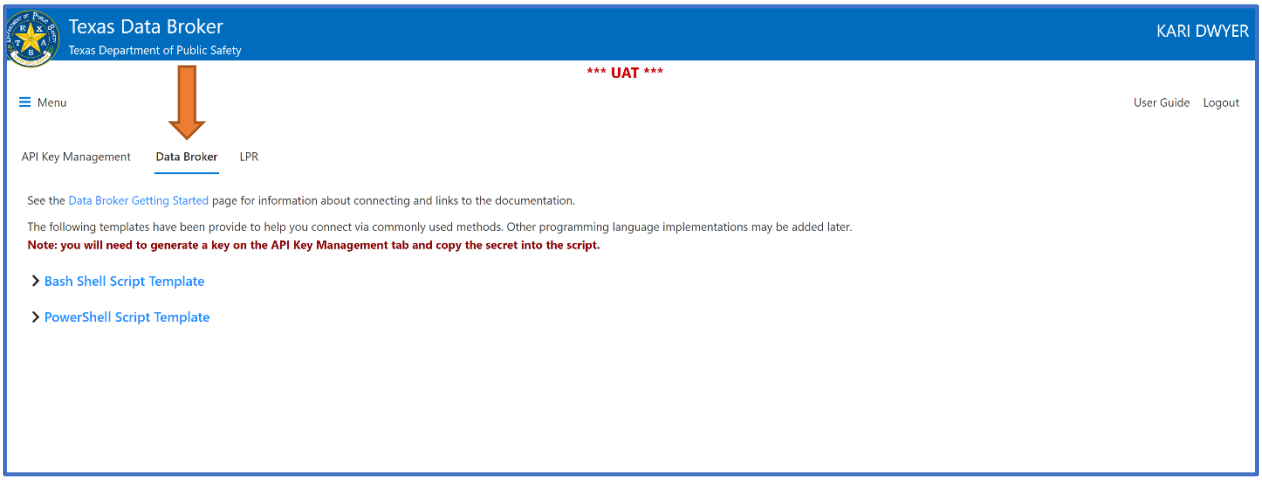
- Clicking **Generate New Key** will display a popup with the Access ID and Secret information:



- Copy the **Access ID** and the **Secret** to a secure location. They will be used in scripts or programs to upload files. The Secret should not be shared or left in an insecure location. You will only be able to copy the Secret one time. If it is lost or compromised, you must generate a new key and de-activate the old one. Multiple keys can be active at a time to enable an entity to transition from one key to another without interruption.

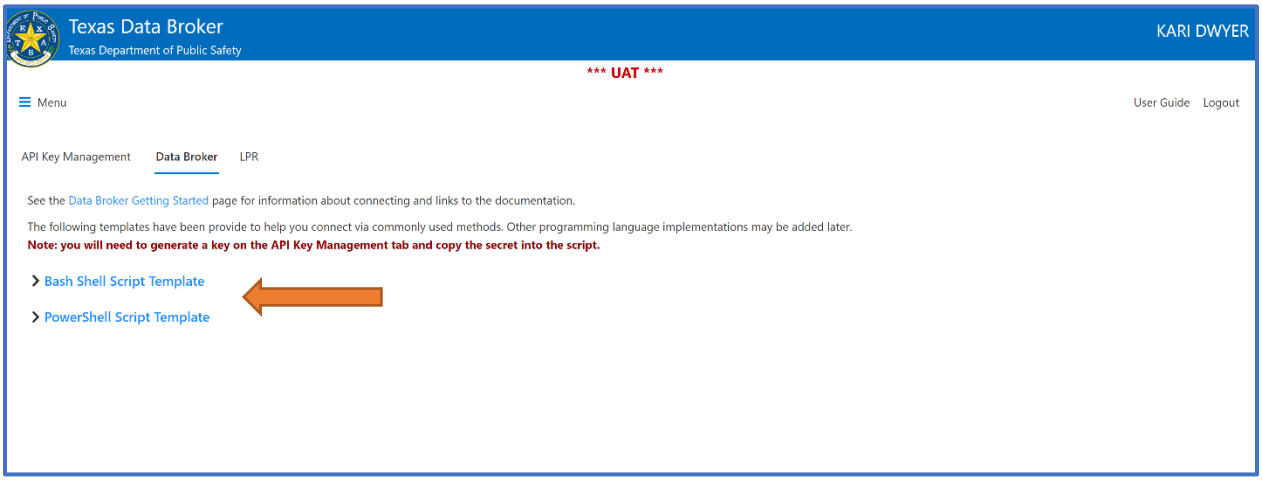
The **Data Broker** tab contains sample templates for submitting via either Linux/Unix or Windows. The Agency is not required to use those templates directly, but they illustrate how the authentication method works.

Below is an example of where to access the Data Broker tab:



To view the templates:

- Click on either the **Bash Shell Script Template** or **PowerShell Script Template**.



2.1.3.4 Uploading Files

This section provides technical details on how to upload a document to the Data Broker, after the Contributing Entity has completed the account setup process and obtained an Access ID and Shared Secret from the website.

The process to submit a file uses an Access ID and a Shared Secret. The Access ID is passed with every request. The Shared Secret is used to create an Authorization header that is used to retrieve a limited-use token. The token can then be used for a period of time to submit documents.

1. An authorized Entity user signs into the Entity Data Broker service account and visits the API Key Management page as described above.

2. When the Entity is ready to submit a file, they must first sign a request to generate a limited use token. They will do this by signing the Shared Secret, the Access ID, and the current time in UTC down to the minute using SHA 512.

   For example, if the secret is ABCDEFGHIJKLMNOP and the Access ID is 1234567890 and the UTC time is 2021-04-08 21:10, the “raw” key would be:

   ABCDEFGHIJKLMNOP1234567890202104082110

   The SHA 512 algorithm would be run on that raw key, resulting in a signed key of:

   4effb797435d5d768994c43e6a560ffecb44dd77d16f585a6edf6af140ce124a35f30a5a5804c499c90a17a37eae5b58b0b2b8166773a13a2b13b3ed1ff439c8

   It is critical that there be no new lines or carriage returns in the raw key and that the date be given in UTC.

   An HTTP POST request for a token must be sent to the token endpoint, passing the Access ID in a query parameter called “accessId” and a header called Authorization with a value of the signed key. During testing, it is also useful to send a header called “Date” with the current time in UTC. The response will be a token that is used to submit a document.

   To submit a document, an HTTP POST request is sent to the submission endpoint, again passing the Access ID in a query parameter called “accessId” and an Authorization header, this time with a value of “Bearer“ followed by the token retrieved above. The submit request should also have a Content-Type header set to text/xml and the body of the POST should be the XML document being submitted.

There are two endpoints for each environment (UAT and Production), differing only in the hostname:

Production:

- *Retrieve the limited use token (described below):* **https://databroker.dps.texas.gov/api/token**
- Submit a document: **https://databroker.dps.texas.gov/api/databroker/submit**

UAT:

- Retrieve the limited use token (described below): **https://databrokeruat.dps.texas.gov/api/token**
- Submit a document: **https://databrokeruat.dps.texas.gov/api/databroker/submit**

Examples are included below for the bash shell on Linux/Unix/OS X and Powershell on Windows. Other programming languages can be used as long as they generate the correct data. **In particular, the key that is hashed below must not include any carriage returns or line feeds and the date must be in UTC.**

Example:

Linux/Unix:

```
# Common variables for every submission (until you generate a new key)
accessId='<accessid>'
# Copy the secret into this script. The one below is just part of it.
# Note: we do not recommend storing the secret in a script
secret='<secret>'
ori="TXDPS0000"
tokenUrl="https://databroker.dps.texas.gov/api/token?accessId=${accessId}"

# Note: make sure to use UTC
date=`date -u +'%Y%m%d'`
time=`date -u +'%H%M'`

# Variables unique to this submission
sequence="0001"
documentId="${ori}_${date}_${time}_${sequence}.xml"
submitUrl="https://databrokeruat.dps.texas.gov/api/databroker/submit?accessId=${accessId}&documentId=${documentId}"

# Get the temporary token
rawKey="${secret}${accessId}${date}${time}"
hashedKey=$(echo -n $rawKey|shasum -a 512|cut -d" " -f1)
headerDate=`date -u '+%a, %d %b %Y %T %Z'`
apiKey=`curl -v -X POST -H "Authorization: ${hashedKey}" -H "Date: ${headerDate}" "${tokenUrl}"`
case "${apiKey}" in error:*)
  echo "Failed to get token: ${apiKey}"
  exit 1
esac
echo "API Key: ${apiKey}"

# Note: set path based on the actual location of the file. The next two lines are for illustration only.
pathToDocument="./${documentId}"
echo "<test>12345</test>" > ${pathToDocument}

# Submit the file
curl -X POST -H "Content-Type: text/xml" -H "Authorization: bearer ${apiKey}" -d @${pathToDocument} "${submitUrl}"
```

Windows Powershell

```
# This script has been tested on PowerShell 5.1
# Common variables for every submission (until you generate a new key)
$accessId = '<accessId>'
# Copy the secret into this script. The one below is just part of it.
# Note: we do not recommend storing the secret in a script
$secret = '<secret>'
$ori = 'TX0000000'
$tokenUrl = "https://databrokeruat.dps.texas.gov/api/token?accessId=${accessId}"

$date = get-date

# Note: make sure to use UTC in the signature key
$timeStamp = $date.ToUniversalTime().ToString('yyyyMMddHHmm')
$headerDate = (get-date $date.ToUniversalTime() -Format r)

# Variables unique to this submission
$sequence = "0001"
$time = (get-date $date -UFormat '%H%M')
$date = (get-date $date -UFormat '%Y%m%d')
$documentId = "${ori}_${date}_${time}_${sequence}.xml"

$submitUrl = "https://databrokeruat.dps.texas.gov/api/databroker/submit?accessId=${accessId}&documentId=${documentId}"

# Get the temporary token
$rawKey = "${secret}${accessId}${timeStamp}"
$stream = [System.IO.MemoryStream]::new()
$writer = [System.IO.StreamWriter]::new(${stream})
$writer.write(${rawKey})
$writer.Flush()
$stream.Position = 0
$hashedKey = (Get-FileHash -InputStream ${stream} -Algorithm SHA512).hash
$headers = @{"Authorization" = "${hashedKey}"; "Date" = "${headerDate}" }
try {
  $apiKey = Invoke-WebRequest -Uri ${tokenUrl} -Method 'POST' -Headers ${headers}
  if (${apiKey} -like 'error:*') {
    echo "Failed to get token: ${apiKey}"
    Return
  }
} catch {
  echo "Failed to get token: $_"
  Return
}

# Note: set path based on the actual location of the file. The next two lines are for illustration only.
$pathToDocument = "${documentId}"
echo "<test>1234</test>" > ${pathToDocument}

# Submit the file
$headers = @{"Content-Type" = "text/xml"; "Authorization" = "bearer ${apiKey}"; }
Invoke-WebRequest -Uri "${submitUrl}" -Method 'POST' -Headers ${headers} -Infile ${pathToDocument}
```

An HTTP response code of 200 from the upload end point can be used to determine if the document was uploaded successfully. If the code is anything else, the file was not submitted successfully. Only the HTTP code is necessary to determine if the upload was successful. However, the response text will be an XML string that indicates the reason for the failure, if possible.

The response text for a successful upload will be:

```
<response>
  <id>…the document ID from the submission…</id>
  <reference>… an internal Data Broker number</reference>
  <time>…the timestamp (ISO8601)</time>
</response>
```

The response text for an upload that fails will be:

```
<response>
 <error>
  <code>400</code>
  <text>reason</text>
 </error>
 <time>…the timestamp (ISO8601)</time>
</response>
```

Inbound data validation occurs when a NIEM document is submitted to the Data Broker process. Validation is not done at the time of submission. The response of the submission only indicates if the Data Broker successfully received the document.

2.1.4 Validation

The Data Broker will validate inbound NIEM IEPD submissions specific to the published IEPD standard available from the Texas Data Broker website.

In the event of a Data Broker system failure, the Contributing Entity may receive an error with a message stating to contact DPS.

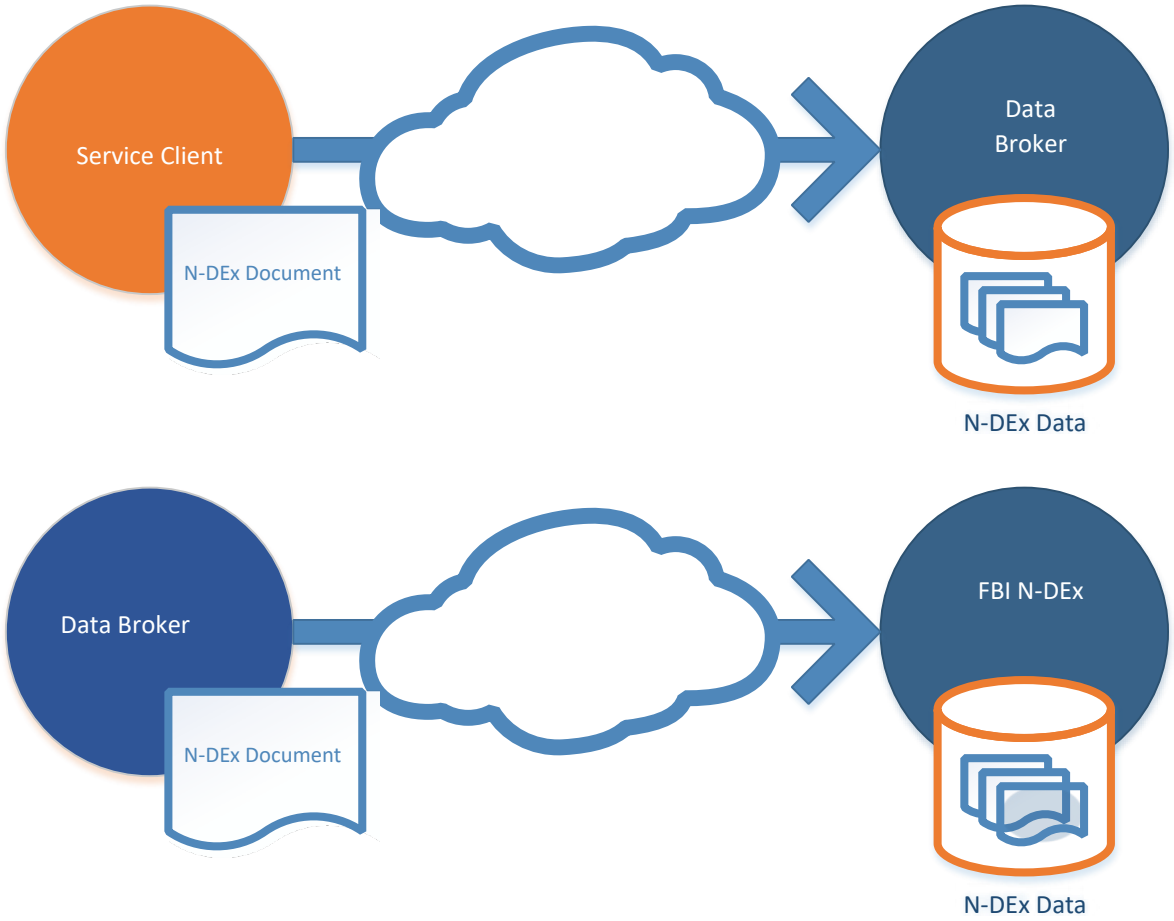
2.2 Workflow

Contributing Entities will use one of the three processes listed below to submit NIEM IEPD documents to the Data Broker via an HTTP Post web service request. All processes use the same API endpoints and authentication mechanism. The HTTP response will indicate whether or not the document was successfully ingested. The following status codes will be returned:

- 200 (OK) – The document was successfully ingested. *Note: this does not mean the document was determined to be valid, only that it was received.*
- 400 (Bad Request) – This indicates that the web service did not understand the request, such as if a parameter was missing.
- 403 (Forbidden) – This indicates that the client was not authorized to submit the document or that the authorization information sent with the request was invalid.
- 500 (Internal Error) – This indicates a problem with the Data Broker. The file may be re-submitted, but if the error continues, DPS should be notified at databroker@dps.texas.gov.

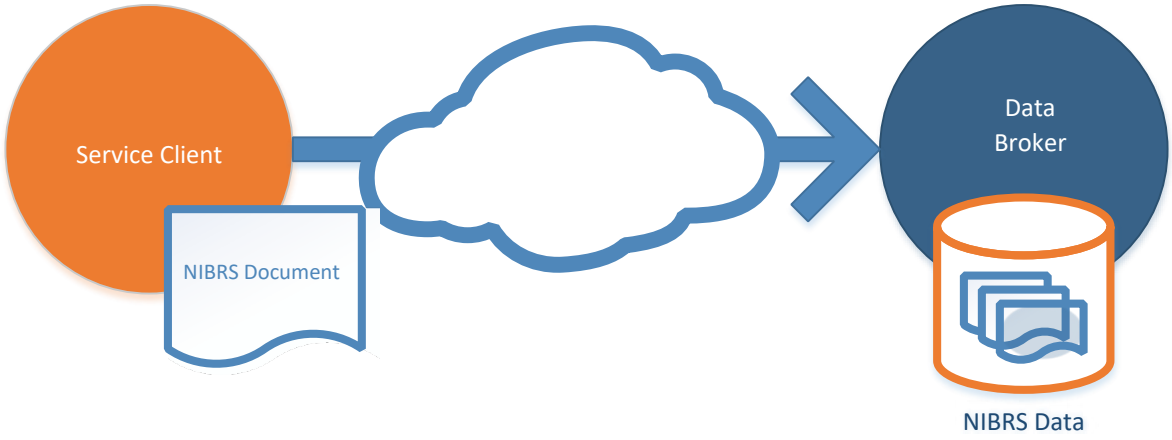
2.2.1 Data Broker Data Submission Workflow for N-DEx Documents

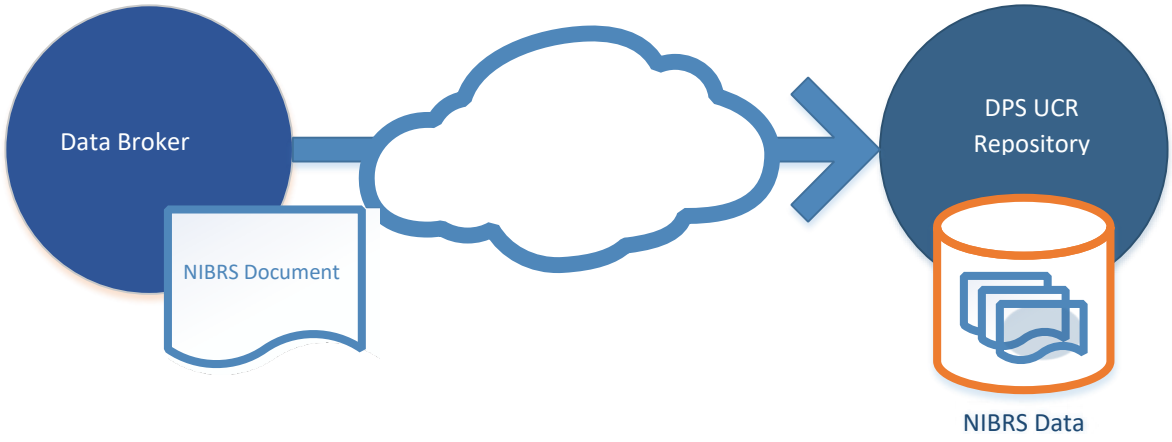
Contributing Entities use the SubmitNDExDocument process to submit NIEM N-DEx IEPD documents to the Data Broker. Each document is initially validated using XSD Schema validation against the N-DEx IEPD. Following successful validation, the Data Broker transmits the document to the FBI N-DEx system.



2.2.2 Data Broker Data Submission Workflow for NIBRS Documents

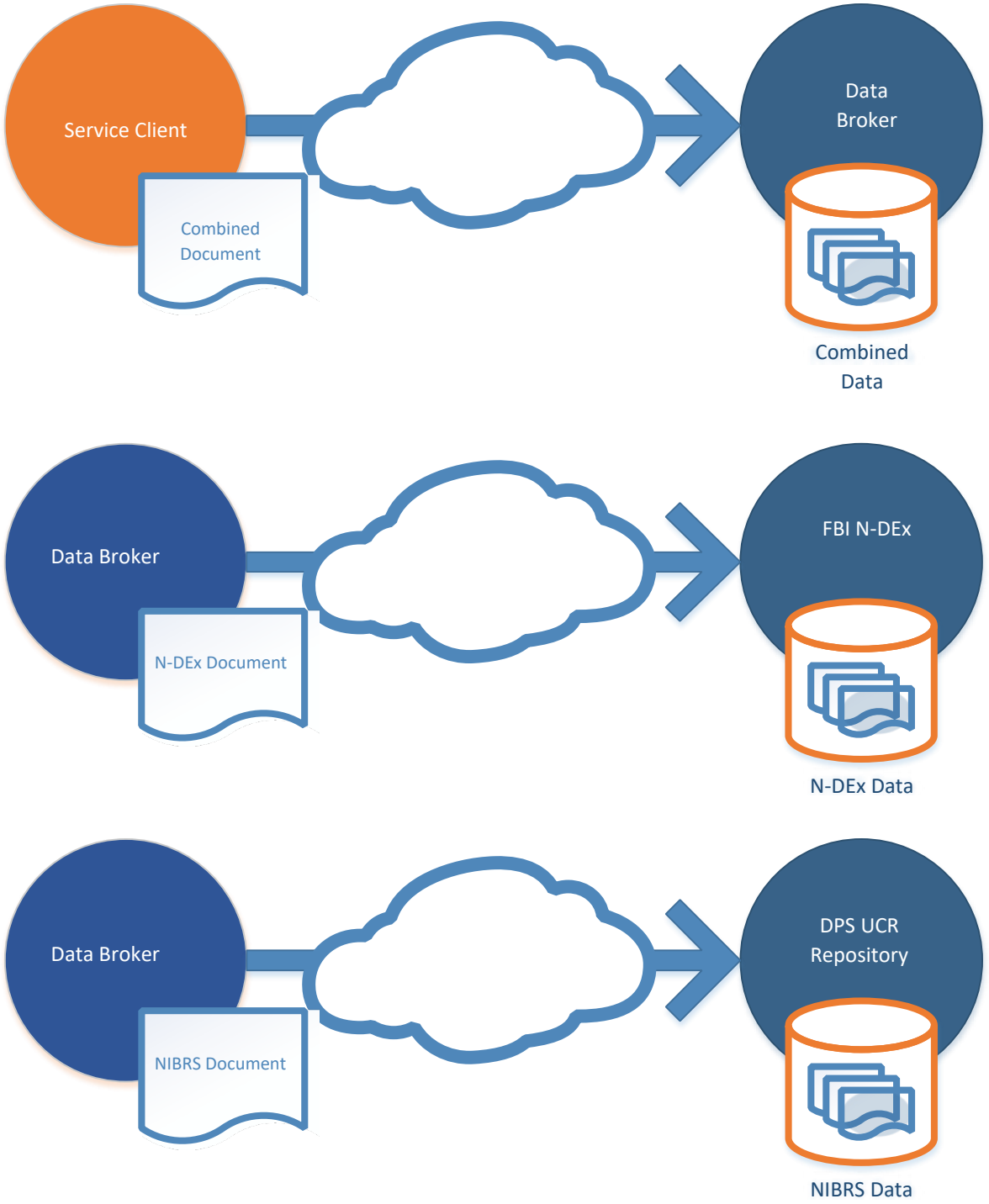
Contributing Entities use the SubmitNIBRSDocument process to submit NIEM NIBRS IEPD documents to the Data Broker. Each document is initially validated using XSD Schema validation. Once a document successfully passes schema validation, a document is then validated using the FBI XCOTA business rules. Following successful submission, the Data Broker transmits the document to the DPS UCR Repository.





2.2.3 Data Broker Data Submission Workflow for N-DEx and NIBRS Documents

Contributing Entities use the SubmitCombinedDocument process to submit NIEM N-DEx IEPD documents to the Data Broker. Each document is initially validated using XSD Schema validation against the N-DEx IEPD. Following successful submission, the Data Broker transmits the document to the FBI N-DEx system and NIBRS information is transmitted to the DPS UCR system for further processing.


Service Client
Combined Document
Data Broker
Combined Data
Data Broker
N-DEx Document
FBI N-DEx
N-DEx Data
Data Broker
NIBRS Document
DPS UCR Repository
NIBRS Data

2.3 Testing and Certification

Submissions from each Contributing Entity will be certified using a test environment prior to being forwarded for testing to N-DEx or DPS UCR.

2.3.1 N-DEx Testing and Certification

The certification process for XML documents containing N-DEx data and formatted using the N-DEx IEPD is as follows:

1. The Contributing Entity creates sample documents for each of the defined N-DEx record types. Once sample XML documents are created, it is recommended that they be validated against the IEPD and against the CJIS XML Conformance Testing Assistant (XCOTA) tool. These validations are used by the Data Broker and by N-DEx.
2. The Contributing Entity submits the documents to the Data Broker test environment using the upload process described above.
3. The Data Broker validates the documents against the N-DEx IEPD xml schemas.
4. The Data Broker validates the documents using the CJIS XCOTA tool.
5. If there are errors, the DPS representative will contact the Entity with details.
6. When each type of document is determined to be valid, DPS will release the documents to N-DEx for further validation.
7. When N-DEx confirms that the sample submissions are valid, the Contributing Entity will be marked as certified and provided access to the production Data Broker environment.
8. The Contributing Entity begins submitting production documents.

2.3.2 NIBRS Testing and Certification

The certification process for XML documents containing NIBRS data and formatted using either the NIBRS IEPD or the N-DEx IEPD is as follows:

1. The Contributing Entity creates sample documents for each of the defined NIBRS record types. Once sample XML documents are created, it is recommended that they be validated against the IEPD and against the CJIS XML Conformance Testing Assistant (XCOTA) tool. These validations are used by the Data Broker.
2. The Contributing Entity submits the documents to the Data Broker test environment using the upload process described above.
3. The Data Broker validates the documents against the NIBRS or N-DEx IEPD xml schemas.
4. The Data Broker forwards the documents to the Texas UCR system.
5. If there are errors, the DPS representative will contact the Entity with details.
6. When the Texas UCR staff confirms that the sample submissions are valid, the Contributing Entity will be marked as certified and provided access to the production Data Broker environment.
7. The Contributing Entity begins submitting production documents.

3 Examples

This section provides examples for utilizing the Data Broker Services. Where applicable, data standards should be deferred to their respective current documentation.

3.1 Data Broker Reporting Data Submission Workflow for N-DEx Documents

3.1.1 SubmitNDExDocument Data XML Example

```
<?xml version="1.0" encoding="UTF-8"?>
<ndex:Submission xsi:schemaLocation="http://fbi.gov/cjis/ndex/4.0 ../base-xsd/ndex/4.0/ndex.xsd...>
 <cjis:MessageMetadata>
  …
 </cjis:MessageMetadata>
 <ndex:IncidentReport>…</ndex:IncidentReport>
 <ndex:Arrest>…</ndex:Arrest>
 … other ndex elements
</ndex:Submission>
```

3.2 Data Broker Reporting Data Submission Workflow for NIRBR Documents

3.2.1 SubmitNIBRSDocument Data XML Example

```
<?xml version="1.0" encoding="UTF-8"?>
<nibrs:Submission xmlns:nibrs="http://fbi.gov/cjis/nibrs/2019.2" …>
 <cjis:MessageMetadata>
 …
 </cjisMessageMetadata>
 <nibrs:Report>
  <nibrs:ReportHeader>
   <!-- Submission Type -->
   <nibrs:NIBRSReportCategoryCode>GROUP A INCIDENT REPORT</nibrs:NIBRSReportCategoryCode>
   <!-- Submission Action Type -->
   <nibrs:ReportActionCategoryCode>I</nibrs:ReportActionCategoryCode>
   …
  <nibrs:ReportHeader>
  <nc:Incident>…see NIBRS 2019.2 IPED for details</nc:Incident>
</nibrs:Submission>
```

3.3 Data Broker Reporting Data Submission Workflow for Combined N-DEx and NIBRS Documents

3.3.1 SubmitCombineDocument Data XML Example

```
<?xml version="1.0" encoding="UTF-8"?>
<ndex:Submission xsi:schemaLocation="http://fbi.gov/cjis/ndex/4.0 ../base-xsd/ndex/4.0/ndex.xsd...>
 <cjis:MessageMetadata>
 …
 </cjis:MessageMetadata>
 <ndex:IncidentReport>
  …
  <ndex:NIBRSReportCategoryCode>GROUP B ARREST REPORT</ndex:NIBRSReportCategoryCode>
  <ndex:NIBRSReportActionCategoryCode>
    A
   </ndex:NIBRSReportActionCategoryCode>
 </ndex:IncidentReport>
 <ndex:Arrest>…</ndex:Arrest>
 … other ndex elements
</ndex:Submission>
```