Using the StreamsLookup error handler

Data Streams Mainnet Access

Chainlink Data Streams is available on Arbitrum Mainnet and Arbitrum Sepolia.Contact us to talk to an expert about integrating Chainlink Data Streams with your applications.

The Chainlink Automation StreamsLookup error handler provides insight into potential errors or edge cases in StreamsLookup upkeeps. The table below outlines a range of error codes and the behavior associated with the codes. Use thecheckErrorHandlerfunction to specify how you want to respond to the error codes.checkErrorHandleris simulated offchain and determines what action for Automation to take onchain inperformUpkeep.

Developer responsibility

Developers implementing Chainlink products are solely responsible for maintaining the security and user experience of their applications. Developers must monitor and mitigate any potential application code risks that may, among other things, result in unanticipated application behavior, including by instituting requisiterisk mitigation processes including, but not limited to, data quality checks, circuit breakers, and appropriate contingency logic for their use case.

Error handler

When Automation detects an event, it runs thecheckLogfunction, which includes aStreamsLookup revert custom error. The StreamsLookup revert enables your upkeep to fetch a report from Data Streams. If reports are fetched successfully, thecheckCallback function is evaluated offchain. Otherwise, thecheckErrorHandlerfunction is evaluated offchain to determine what Automation should do next. Both of these functions have the same output types (bool upkeepNeeded, bytes memory performData), which Automation uses to runperformUpkeeponchain. Theexample code also shows each function outlined in the diagram below:

If the Automation network fails to get the requested reports, an error code is sent to thecheckErrorHandlerfunction in your contract. If your contract doesn't have thecheckErrorHandlerfunction, nothing will happen. If your contract has thecheckErrorHandlerfunction, it is evaluated offchain to determine what to do next. For example, you could intercept or ignore certain errors and decide not to runperformUpkeepin those cases, in order to save time and gas. For other errors, you can execute an alternative path withinperformUpkeep, and the upkeep runs the custom logic you define in yourperformUpkeepfunction to handle those errors.

1. Add thecheckErrorHandlerfunction in your contract to specify how you want to handleerror codes . For example, you could decide to ignore any codes related to bad requests or incorrect input, without runningperformUpkeeponchain:

functioncheckErrorHandler(uinterrorCode,bytescalldataextraData)externalreturns(boolupkeepNeeded,bytesmemoryperformData){// Add custom logic to handle errors offchain herebool_upkeepNeeded=true;boolsuccess=false;boolisError=true;if(errorCode==808400){// Handle bad request error code offchain_upkeepNeeded=false;}else{// logic to handle other errors}return(_upkeepNeeded,abi.encode(isError,abi.encode(errorCode,extraData,success)));} 2. Define custom logic for the alternative path withinperformUpkeep, to handle any error codes you did not intercept offchain incheckErrorHandler:

// function will be performed on-chainfunctionperformUpkeep(bytescalldataperformData)external{// Decode incoming performData(boolisError,bytespayload)=abi.decode(performData)// Unpacking the errorCode from checkErrorHandlerif(isError){(uinterrorCode,bytesmemoryextraData,boolreportSuccess)=abi.decode(payload,(uint,bytes,bool)// Define custom logic here to handle error codes onchain);}else{// Otherwise unpacking info from checkCallback(bytes[]memorysignedReports,bytesmemoryextraData,boolreportSuccess)=abi.decode(payload,(bytes[],bytes,bool));if(reportSuccess){bytesmemoryreport=signedReports[0];(,bytesmemoryreportData)=abi.decode(report,(bytes32[3],bytes));// Logic to verify and decode report}else{// Logic in case reports were not pulled successfully}}}

Testing checkErrorHandler

checkErrorHandleris simulated offchain. WhenupkeepNeededreturnstrue, Automation runsperformUpkeeponchain using theperformDatafromcheckErrorHandler. If thecheckErrorHandlerfunction itself reverts,performUpkeepdoes not run.

If you need to force errors in StreamsLookup while testing, you can try the following methods:

- Specifying an incorrectfeedIDto force error code 808400 (ErrCodeStreamsBadRequest)
- Specifying a future timestamp to force error code 808206 (where partial content is received) for both singlefeedIDand bulkfeedIDrequests
- Specifying old timestamps for reports not available anymore yields either error code 808504 (no response) or 808600 (bad response), depending on which service calls the timeout request

If yourStreamsLookup revert function is defined incorrectly in your smart contracts, the nodes will not be able to decode it.

Error codes

Error codeRetriesPossible cause of errorNo errorN/ANo errorErrCodeStreamsBadRequest: 808400NoUser requested 0 feedsUser error, incorrect parameter inputIssue with encoding http url (bad characters)ErrCodeStreamsUnauthorized: 808401NoKey access issue or incorrect feedID808206Log trigger - after retries; Conditional immediatelyRequested m reports but only received n (partial)8085XX (e.g 808500)Log trigger - after retries; Conditional immediatelyNo responseErrCodeStreamsBadResponse: 808600NoError in reading body of returned response, but service is upErrCodeStreamsTimeout: 808601NoNo valid report is received for 10 secondsErrCodeStreamsUnknownError: 808700NoUnknown

Example code

This example code includes therevert StreamsLookup,checkCallback,checkErrorHandlerandperformUpkeepfunctions. The full code example is availablehere .

```
// SPDX-License-Identifier:
MITpragmasolidity^0.8.16;import{StreamsLookupCompatibleInterface}from"@chainlink/contracts/src/v0.8/automation/interfaces/StreamsLookupCompatibleInterface.sol";import{ILogAutomation,Log}from"
feeds/interfaces/IRewardManager.sol";import{IVerifierFeeManager}from"@chainlink/contracts/src/v0.8/llo-
feeds/interfaces/IVerifierFeeManager.sol";import{IERC20}from"@chainlink/contracts/src/v0.8/vendor/openzeppelin-
solidity/v4.8.0/contracts/interfaces/IERC20.sol";import{Common}from"@chainlink/contracts/src/v0.8/libraries/Common.sol";/* * THIS IS AN EXAMPLE CONTRACT THAT USES UN-AUDITED CODE. *
DO NOT USE THIS CODE IN PRODUCTION.
////////////////////////////////////////////////////////////////INTERFACES////////////////////////////////////interfaceIFeeManager{functiongetFeeAndReward(addresssubscriber,bytesmemoryreport,addressquoteAddress)externalreturns(C
INTERFACES////////////////////////////////////contractStreamsLookupChainlinkAutomationisILogAutomation,StreamsLookupCompatibleInterface{structBasicReport{bytes32feedId;// The feed ID the report has
data foruint32validFromTimestamp;// Earliest timestamp for which price is applicableuint32observationsTimestamp;// Latest timestamp for which price is applicableuint192nativeFee;// Base cost to
validate a transaction using the report, denominated in the chain’s native token (WETH/ETH)uint192linkFee;// Base cost to validate a transaction using the report, denominated in
LINKuint32expiresAt;// Latest timestamp where the report can be verified on-chainint192price;// DON consensus median price, carried to 18 decimal places}structPremiumReport{bytes32feedId;// The
feed ID the report has data foruint32validFromTimestamp;// Earliest timestamp for which price is applicableuint32observationsTimestamp;// Latest timestamp for which price is
applicableuint192nativeFee;// Base cost to validate a transaction using the report, denominated in the chain’s native token (WETH/ETH)uint192linkFee;// Base cost to validate a transaction using the
report, denominated in LINKuint32expiresAt;// Latest timestamp where the report can be verified on-chainint192price;// DON consensus median price, carried to 18 decimal placesint192bid;// Simulated
price impact of a buy order up to the X% depth of liquidity utilisationint192ask;// Simulated price impact of a sell order up to the X% depth of liquidity
utilisation}structQuote{addressquoteAddress;}eventPriceUpdate(int192indexedprice);IVerifierProxypublicverifier;addresspublicFEE_ADDRESS;stringpublicconstantSTRING_DATASTREAMS_FEEDLABI
["0x00027bbaff688c906a3e20a34fe951715d1018d262a5b66e38eda027a674cd1b"// Ex. Basic ETH/USD price report];constructor(address_verifier){verifier=IVerifierProxy(_verifier);//Arbitrum Sepolia:
0x2ff010debc1297f19579b4246cad07bd24f2488a}functioncheckLog(Logcalldatalog,bytesmemory)externalreturns(boolupkeepNeeded,bytesmemoryperformData)
{revertStreamsLookup(STRING_DATASTREAMS_FEEDLABEL,feedIds,STRING_DATASTREAMS_QUERYLABEL,log.timestamp,"");}functioncheckCallback(bytes[]calldatavalues,bytescalldataextraDat
{bool_upkeepNeeded=true;boolsuccess=true;boolisError=false;return(_upkeepNeeded,abi.encode(isError,abi.encode(values,extraData,success)));}functioncheckErrorHandler(uinterrorCode,bytescalldat
{bool_upkeepNeeded=true;boolsuccess=false;boolisError=true;// Add custom logic to handle errors offchain hereif(errorCode==808400){// Bad request error code_upkeepNeeded=false;}else{// logic to
handle other errors}return(_upkeepNeeded,abi.encode(isError,abi.encode(errorCode,extraData,success)));}// function will be performed on-
chainfunctionperformUpkeep(bytescalldataperformData)external{// Decode incoming performData(boolisError,bytesmemorypayload)=abi.decode(performData,(bool,bytes));// Unpacking the errorCode
from checkErrorHandlerif(isError){(uinterrorCode,bytesmemoryextraData,boolreportSuccess)=abi.decode(payload,(uint,bytes,bool));// Custom logic to handle error codes onchain}else{// Otherwise
unpacking info from checkCallback(bytes[]memorysignedReports,bytesmemoryextraData,boolreportSuccess)=abi.decode(payload,(bytes[],bytes,bool));if(reportSuccess)
{bytesmemoryreport=signedReports[0];(,bytesmemoryreportData)=abi.decode(report,(bytes32[3],bytes));// BillingIFeeManager
feeManager=IFeeManager(address(verifier.s_feeManager()));IRewardManager
rewardManager=IRewardManager(address(feeManager.i_rewardManager()));addressfeeTokenAddress=feeManager.i_linkAddress();
(Common.Assetmemoryfee,,)=feeManager.getFeeAndReward(address(this),reportData,feeTokenAddress);IERC20(feeTokenAddress).approve(address(rewardManager),fee.amount);// Verify the
reportbytesmemoryverifiedReportData=verifier.verify(report,abi.encode(feeTokenAddress));// Decode verified report data into BasicReport
structBasicReportmemoryverifiedReport=abi.decode(verifiedReportData,(BasicReport));// Log price from reportemitPriceUpdate(verifiedReport.price);}else{// Logic in case reports were not pulled
successfully}}}fallback()externalpayable{}
```

Open in Remix What is Remix?