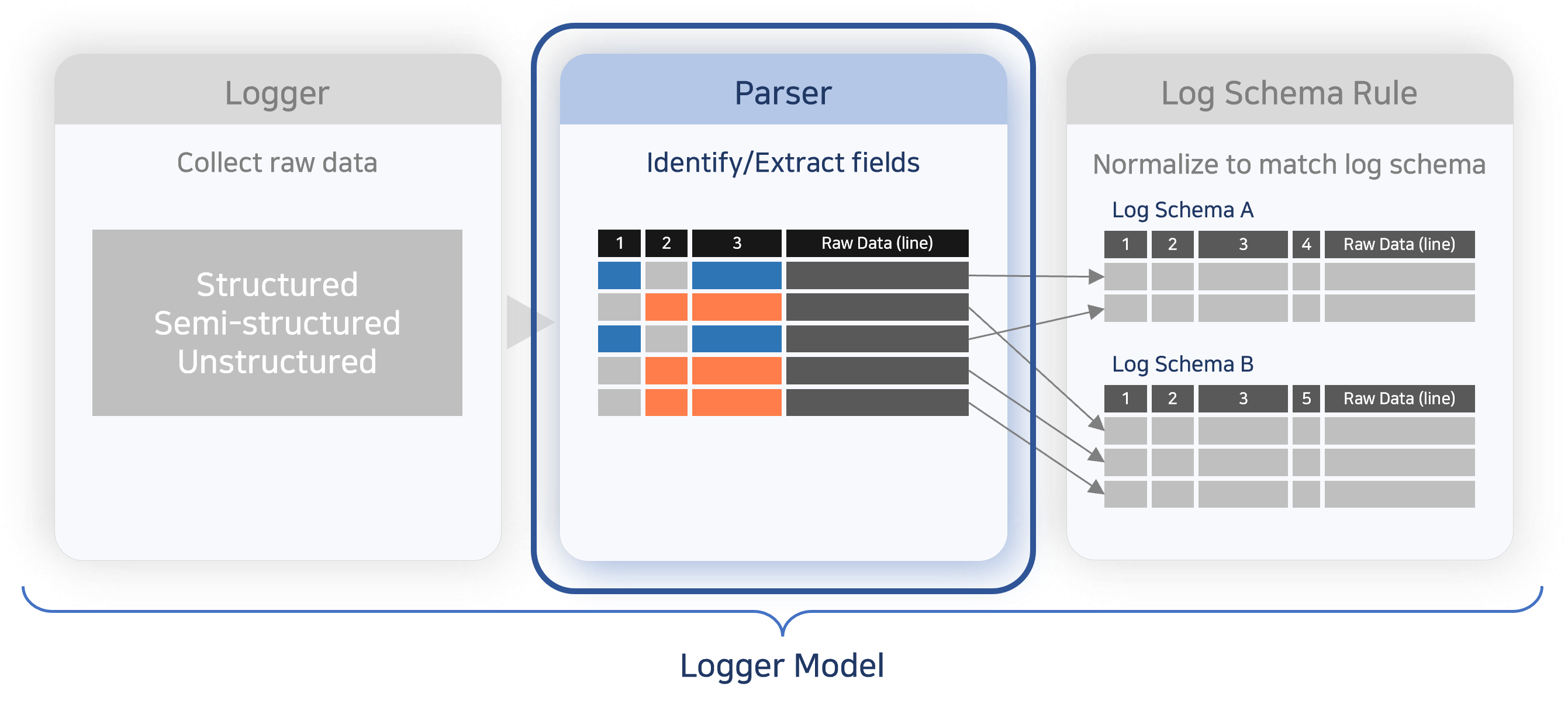
### パーサー

#### 概要

パーサーは、入力データをレコード単位で識別し、各レコードをフィールド単位で分割します。



パーサーの主な特徴は以下の通りです。

**元レコードの保持:**

パーサーは元のレコードを保持するように設計されています。元データ自体を加工するのではなく、そこからフィールドを識別・抽出し、個別のフィールドとして生成します。ログプレッソ・ソナーのクエリでは、通常**line**フィールドに元レコードが出力されます。

**ログフォーマットごとの専用パーサー**

同一種別のシステムであっても、メーカーごとにログフォーマットが異なります。例えば、Palo Alto NetworksのファイアウォールログはCSV形式、Fortinet FortiGateのファイアウォールログはスペース区切りのkey=value形式です。ログプレッソ・ソナーでは、各ログフォーマットに特化した専用パーサーを用いて元ログからフィールド値を抽出します。

以下は、Palo Alto Networksファイアウォールログの元レコード例です。

Apr 10 04:38:54 1,2012/04/10 04:38:54,012345678911,THREAT,spyware,1,2012/04/10 04:38:49,192.0.2.255,192.0.2.2,0.0.0.0,0.0.0.0,rule1,,crusher,web-browsing,vsys1,untrust,trust,ethernet1/2,ethernet1/1,forwardAll,2012/04/10 04:38:54,25466,1,80,59127,0,0,0x200000,tcp,drop-all-packets,"habl.bin",Trojan-Spy.Win32.Zbot.wti(12620),any,medium,server-to-client,0,0x0,United States,192.0.2.0-192.0.2.255,0,

上記ログのパース結果を確認するには、ログプレッソストアから[Palo Alto Networks Firewall](https://logpresso.store/en/apps/paloalto-ngfw)アプリをダウンロード・インストールし、**分析** > **クエリ**で以下のクエリを実行してください。

json "{}" | eval line="Apr 10 04:38:54 1,2012/04/10 04:38:54,012345678911,THREAT,spyware,1,2012/04/10 04:38:49,192.0.2.255,192.0.2.2,0.0.0.0,0.0.0.0,rule1,,crusher,web-browsing,vsys1,untrust,trust,ethernet1/2,ethernet1/1,forwardAll,2012/04/10 04:38:54,25466,1,80,59127,0,0,0x200000,tcp,drop-all-packets,\"habl.bin\",Trojan-Spy.Win32.Zbot.wti(12620),any,medium,server-to-client,0,0x0,United States,192.0.2.0-192.0.2.255,0," | parse overlay=t paloalto-ngfw

次に、Fortinet FortiGateファイアウォールログの元レコード例です。

date=2024-05-27 time=00:00:08 devname=FGT01234567890AB devid=FGT01234567890AB logid=0000000013 type=traffic subtype=forward level=notice vd=public srcip=198.51.100.150 srcport=39986 srcintf="npu0\_vlink0" dstip=1.1.1.1 dstport=53 dstintf="wan1" poluuid=aa2401fa-d101-51e8-1c53-c8700baa2214 sessionid=26168801 proto=17 action=accept policyid=9 policytype=policy dstcountry="Korea, Republic of" srccountry="Reserved" trandisp=snat transip=203.0.113.2 transport=39986 service="DNS" duration=180 sentbyte=58 rcvdbyte=138 sentpkt=1 rcvdpkt=1 appcat="unscanned" date=2024-05-27 time=00:00:16 devname=FGT01234567890AB devid=FGT01234567890AB logid=0814044032 type=utm subtype=voip eventtype=voip level=information vd=root session\_id=194 epoch=0 event\_id=162474 srcip=198.51.100.151 src\_port=5060 dstip=192.0.2.56 dst\_port=5060 proto=17 src\_int="internal" dst\_int="npu0\_vlink1" policy\_id=1 profile="default" voip\_proto=sip kind=register action=permit status=succeeded duration=0 dir=outbound call\_id="13dcef6f@192.168.0.3" from="sip:3049@voip.example.com:5060" to="sip:3049@voip.example.com:5060"

上記ログのパース結果を確認するには、ログプレッソストアから[FortiGate](https://logpresso.store/en/apps/fortigate)アプリをダウンロード・インストールし、**分析** > **クエリ**で以下のクエリを実行してください。

json "[ {'line': 'date=2024-05-27 time=00:00:08 devname=FGT01234567890AB devid=FGT01234567890AB logid=0000000013 type=traffic subtype=forward level=notice vd=public srcip=198.51.100.150 srcport=39986 srcintf=\"npu0\_vlink0\" dstip=1.1.1.1 dstport=53 dstintf=\"wan1\" poluuid=aa2401fa-d101-51e8-1c53-c8700baa2214 sessionid=26168801 proto=17 action=accept policyid=9 policytype=policy dstcountry=\"Korea, Republic of\" srccountry=\"Reserved\" trandisp=snat transip=203.0.113.2 transport=39986 service=\"DNS\" duration=180 sentbyte=58 rcvdbyte=138 sentpkt=1 rcvdpkt=1 appcat=\"unscanned\"'}, {'line': 'date=2024-05-27 time=00:00:16 devname=FGT01234567890AB devid=FGT01234567890AB logid=0814044032 type=utm subtype=voip eventtype=voip level=information vd=root session\_id=194 epoch=0 event\_id=162474 srcip=198.51.100.151 src\_port=5060 dstip=192.0.2.56 dst\_port=5060 proto=17 src\_int=\"internal\" dst\_int=\"npu0\_vlink1\" policy\_id=1 profile=\"default\" voip\_proto=sip kind=register action=permit status=succeeded duration=0 dir=outbound call\_id=\"13dcef6f@192.168.0.3\" from=\"sip:3049@voip.example.com:5060\" to=\"sip:3049@voip.example.com:5060\"'} ]" | parse fortigate-ngfw

このように、パーサーは様々なデータを標準化フォーマットへ変換し、正規化を実現します。

パーサーの種類：基本パーサーとアプリパーサー

ログプレッソにおけるパーサーは、基本パーサーとアプリパーサーに分類されます。

**アプリパーサー**

多くのパーサーは[アプリ](https://docs.logpresso.comnull)のインストール時に提供されます。アプリパーサーは、対象システムから収集したデータのフィールド抽出に必要な設定があらかじめ構成されており、追加設定はほとんど不要です。

アプリパーサーを変更または削除すると、アプリの動作に支障をきたす場合がありますのでご注意ください。

**基本パーサー**

基本パーサーはクラスター管理者が定義します。ログプレッソ・ソナーをインストールした直後は、[パーサーリスト](https://docs.logpresso.comnull)には何も表示されません。アプリパーサーで対応できないデータ処理が必要な場合、クラスター管理者が基本パーサーを作成して利用できます。

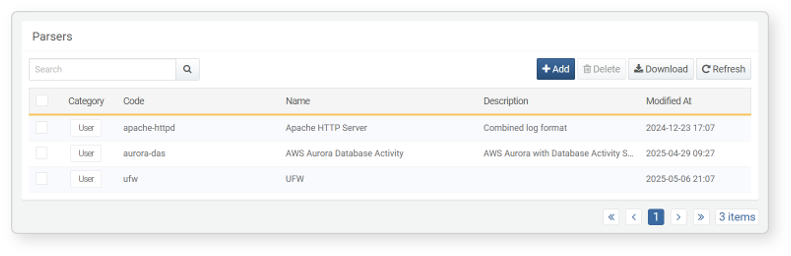
ノードでのパーサー実行

パーサーの実行場所は、ログプレッソ・ソナーの[クラスター](https://docs.logpresso.comnull)構成によって異なります。下表の通り、パーサーはデータがテーブルに書き込まれるノードで実行されます。

|  |  |  |
| --- | --- | --- |
| クラスター | ノード構成（各ノードは冗長化対応） | パーサー実行場所 |
| 1層 | コントロールノード | コントロールノード |
| 2層 | データノード → コントロールノード | データノード |
| 3層 | フォワーダーノード → データノード → コントロールノード | データノード |

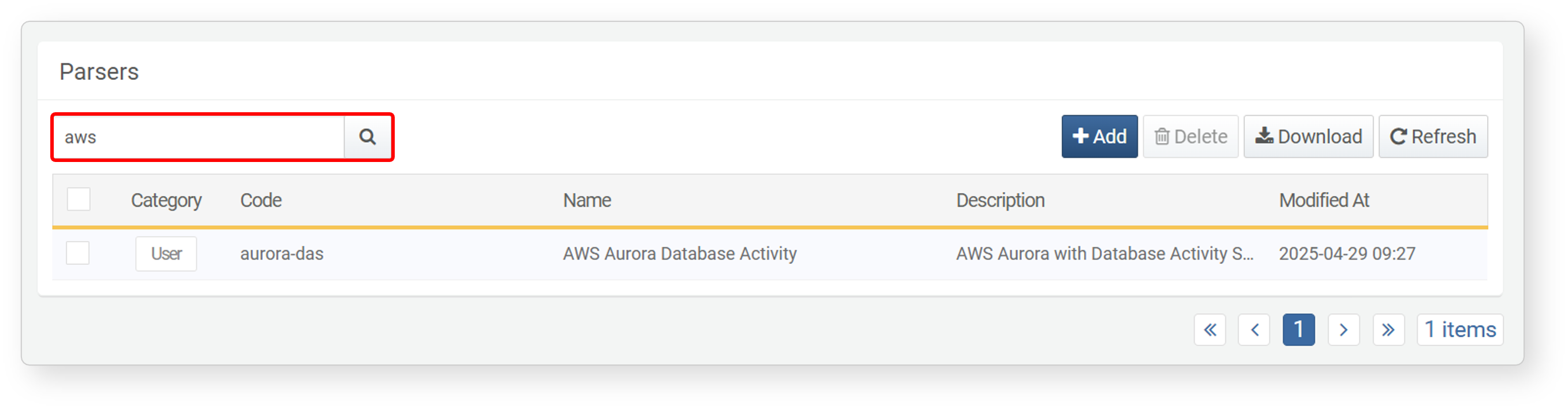
#### パーサーの検索

**ロガー > パーサー**でパーサーの閲覧・管理が可能です。下図は[Palo Alto Networks NGFW](https://logpresso.store/en/apps/paloalto-ngfw)、[BLUEMAX NGF](https://logpresso.store/en/apps/bluemax-ngf)、[Ahnlab Trusguard](https://logpresso.store/en/apps/ahnlab-trusguard)アプリをインストール後のパーサーリスト例です。



* **コード**: クエリ文で利用する一意の識別子
* **名称**: パーサーを識別しやすいユーザー向け名称
* **説明**: パーサーの詳細情報
* **最終更新日**: 作成または最終変更日時

特定のパーサーを探す場合は、ツールバーの検索機能を利用します。検索は**コード**、**名称**、**説明**フィールドに入力したキーワードを含むパーサーを対象とし、大文字・小文字は区別されません。

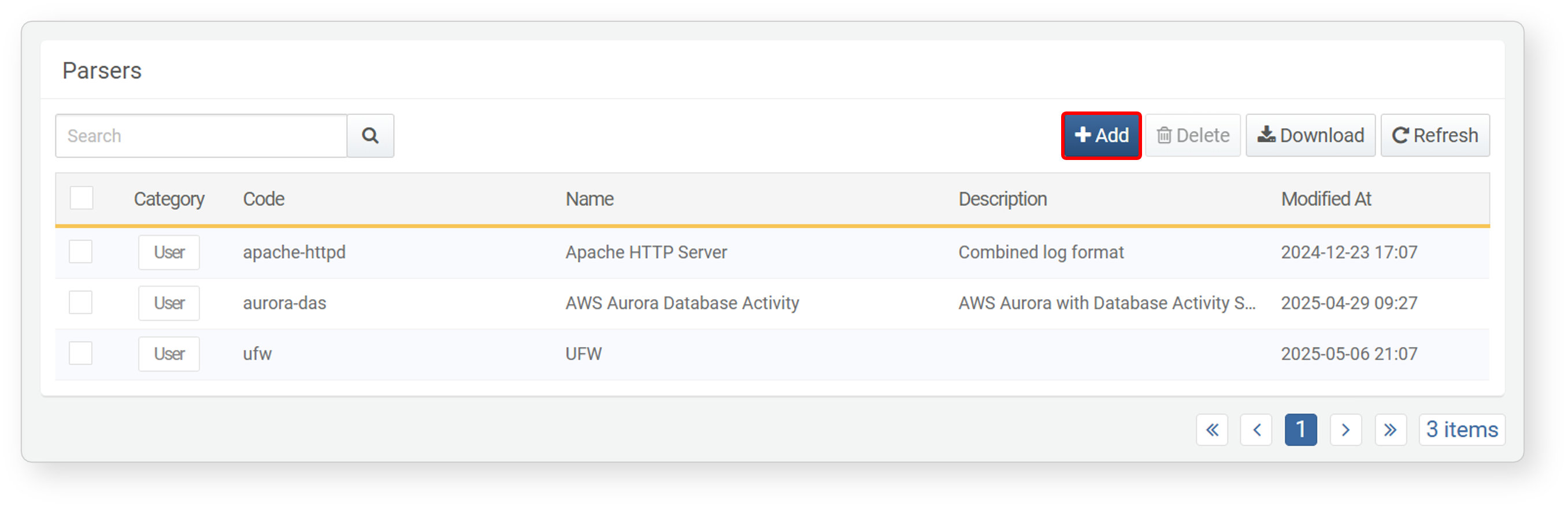


#### パーサーの追加

多くの環境ではアプリパーサーで十分ですが、データソースに対応するアプリパーサーが存在しない場合や、既存パーサーが要件に合わない場合はカスタムパーサーを作成できます。

パーサー追加手順：

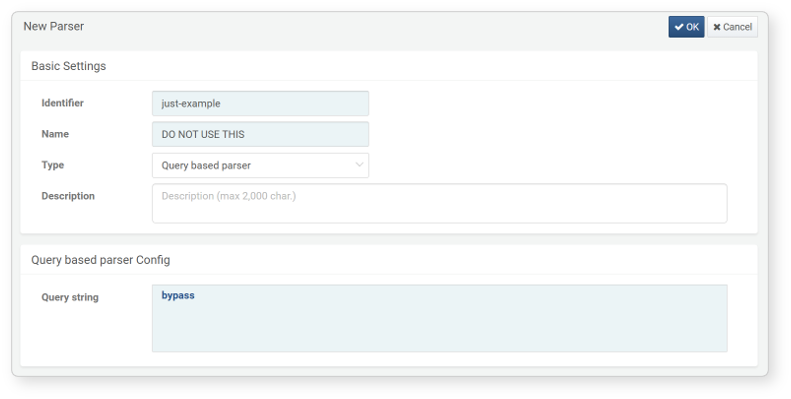
**ロガー > パーサー**に移動し、ツールバーの**追加**ボタンをクリックします。



**パーサー追加**ダイアログで、**基本設定**(section-parsers#outline7)および[パーサー設定](https://docs.logpresso.comnull)に必要事項を入力し、右上の**OK**をクリックします。

基本設定

このセクションでは、パーサーの**識別子**、**名称**、**タイプ**、**説明**などの基本属性を指定します。選択した**タイプ**によっては追加プロパティの入力が必要です。



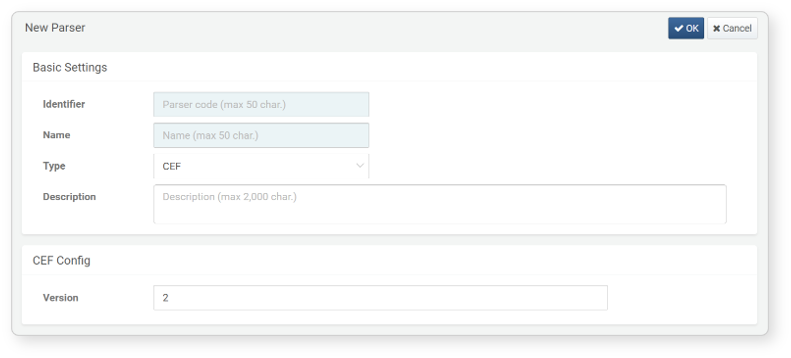
* **識別子**: クエリ文で利用する一意のパーサー識別子。他のパーサーと重複しないようにしてください。
* **名称**: Webコンソール上で識別しやすいパーサー名。
* **タイプ**: 入力データのフォーマットまたはパース方式。デフォルトは[クエリベースパーサー](https://docs.logpresso.comnull)です。
* タイプによって追加設定項目が異なります（詳細は[パーサー設定](https://docs.logpresso.comnull)参照）。
* **説明**: （任意）パーサーの機能や用途に関する補足情報。

パーサー設定

このセクションでは、[基本パーサー](https://docs.logpresso.comnull)各タイプの設定プロパティについて説明します。アプリパーサーの詳細はログプレッソストアの各アプリドキュメントを参照してください。

CEF

CEF（Common Event Format）パーサーは、ArcSightなどのESMやSIEMシステムで広く利用されるCEFフォーマットのデータを処理します。

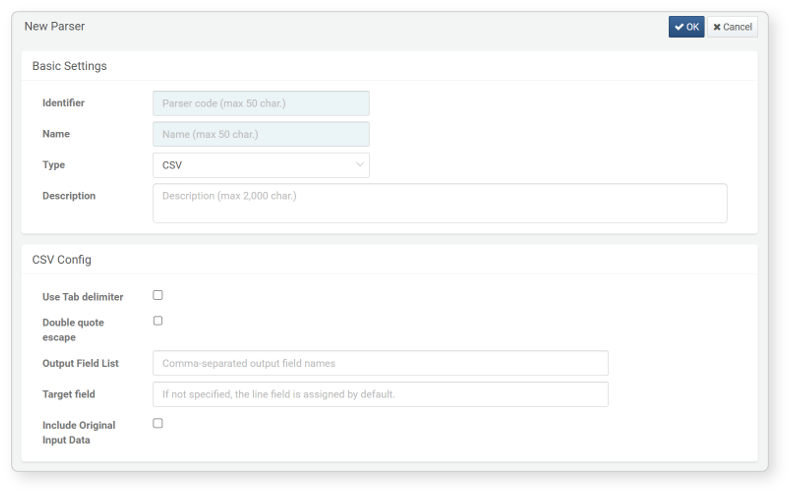


* **バージョン**: パーサーバージョン（デフォルト: 2）。バージョン1は後方互換性のために提供されています。新規CEFパーサーには高度なフィールド正規化をサポートするバージョン2の利用を推奨します。
* **1**: CEF仕様バージョン23に準拠してログをパースします（参考: [CEFログフォーマット（韓国語）](https://ko.logpresso.com/en/blog/2017-03-25-cef-log-format)）。
* **2**: CEF仕様バージョン25に準拠し、下記のようなフィールド正規化をサポートします。

|  |  |  |
| --- | --- | --- |
| 元フィールド名（正規化前） | 正規化後フィールド名 | 型 |
| src | src\_ip | IPアドレス |
| spt | src\_port | 32ビット整数 |
| dst | dst\_ip | IPアドレス |
| dpt | dst\_port | 32ビット整数 |
| sourceTranslatedAddress | nat\_src\_ip | IPアドレス |
| sourceTranslatedPort | nat\_src\_port | 32ビット整数 |
| destinationTranslatedAddress | nat\_dst\_ip | IPアドレス |
| destinationTranslatedPort | nat\_dst\_port | 32ビット整数 |
| dvc | device\_ip | IPアドレス |
| dvchost | device\_host | 文字列 |
| dvcmac | device\_mac | 文字列 |
| dvcpid | device\_pid | 64ビット整数 |
| sourceDnsDomain | src\_domain | 文字列 |
| sourceServiceName | src\_service | 文字列 |
| destinationDnsDomain | dst\_domain | 文字列 |
| destinationServiceName | dst\_service | 文字列 |
| shost | src\_host | 文字列 |
| smac | src\_mac | 文字列 |
| dhost | dst\_host | 文字列 |
| dmac | dst\_mac | 文字列 |
| suid | src\_user\_id | 文字列 |
| suser | src\_user | 文字列 |
| duid | dst\_user\_id | 文字列 |
| duser | dst\_user | 文字列 |
| spid | src\_pid | 64ビット整数 |
| sproc | src\_process | 文字列 |
| dpid | dst\_pid | 64ビット整数 |
| dproc | dst\_process | 文字列 |
| fileCreateTime | file\_ctime | 日時 |
| fileModificationTime | file\_mtime | 日時 |
| fname | file\_name | 文字列 |
| filePath | file\_path | 文字列 |
| fileType | file\_type | 文字列 |
| fsize | file\_size | 64ビット整数 |
| fileHash | file\_hash | 文字列 |
| in | rcvd\_bytes | 64ビット整数 |
| out | sent\_bytes | 64ビット整数 |
| proto | protocol | 文字列 |
| app | app | 文字列 |
| act | action | 文字列 |
| cat | category | 文字列 |
| cnt | event\_count | 32ビット整数 |

CSV

CSV（カンマ区切り値）またはTSV（タブ区切り値）パーサーは、CSVまたはTSV形式のデータを処理します。区切り文字（カンマまたはタブ）で識別された各トークンを**フィールド名**に順番に対応させ、各フィールドに割り当てます。

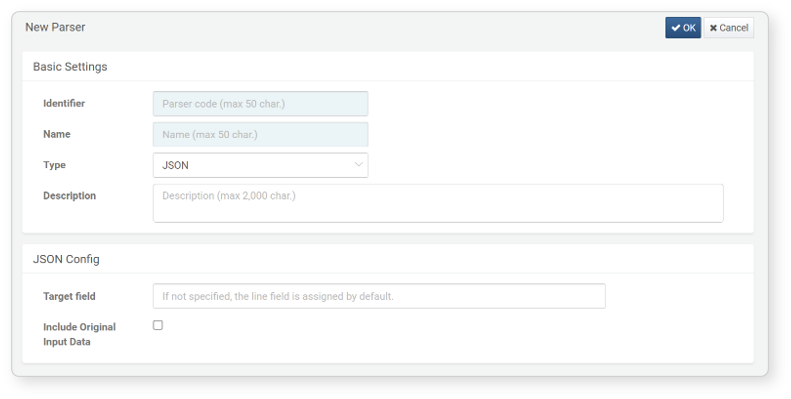


* **タブ区切りを使用**：カンマ（,）の代わりにタブ文字を区切り文字として使用するかどうか（デフォルト：未選択）
* **ダブルクォートエスケープ**：特殊文字のエスケープにダブルクォートを使用するかどうか（デフォルト：未選択）
* **出力フィールドリスト**：解析されたフィールドに順番に割り当てるフィールド名のカンマ区切りリスト（デフォルト：なし、最大10,000文字）。指定しない場合、トークンはcolumn0、column1、...、columnNに割り当てられます。
* **対象フィールド**：解析対象となる入力フィールド名（デフォルト：なし、最大500文字）。指定しない場合、**line**フィールドのデータが解析されます。
* **元の入力データを含める**：解析結果とともに元のフィールド値を含めるかどうか（デフォルト：未選択）。選択した場合、元データは**line**フィールドに含まれます。

CSVパーサーは、カンマ（,）またはタブで区切られた各トークンをcolumn0、column1、...、columnNフィールドに順番に割り当てます。これらのフィールド名は、**フィールド名**で指定された名前に置き換えられます。フィールドリストに10個の名前があり、トークンが13個ある場合、11番目から13番目のトークンはcolumn10、column11、column12に割り当てられます。トークン数が不足している場合、不足分のフィールドにはnullが設定されます。

JSON

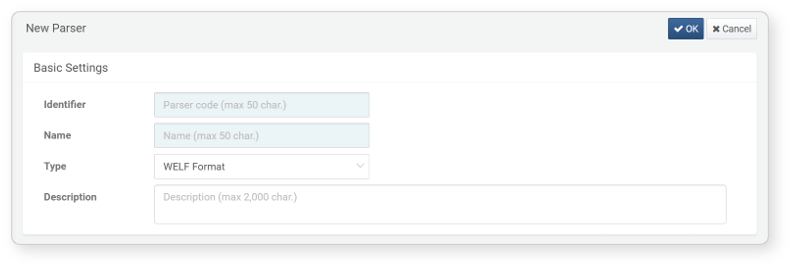
JSONパーサーは、JSON形式のデータを処理します。入力データにネストされたJSONが含まれている場合、ネストされたJSONデータはマップまたは配列形式の複合フィールドに割り当てられます。



* **対象フィールド**：解析対象となる入力フィールド名（デフォルト：なし、最大255文字）。指定しない場合、**line**フィールドのデータが解析されます。
* **元の入力データを含める**：解析結果とともに元のフィールド値を含めるかどうか（デフォルト：未選択）。選択した場合、元データは**line**フィールドに含まれます。

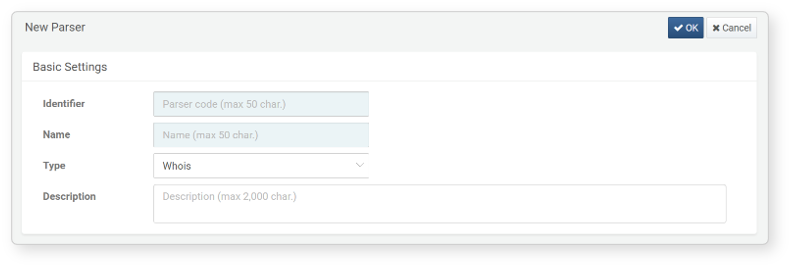
WELF

WELF（WebTrends Enhanced Log Format）パーサーは、WELF形式のデータを処理します。このパーサーには追加で指定する属性はありません。



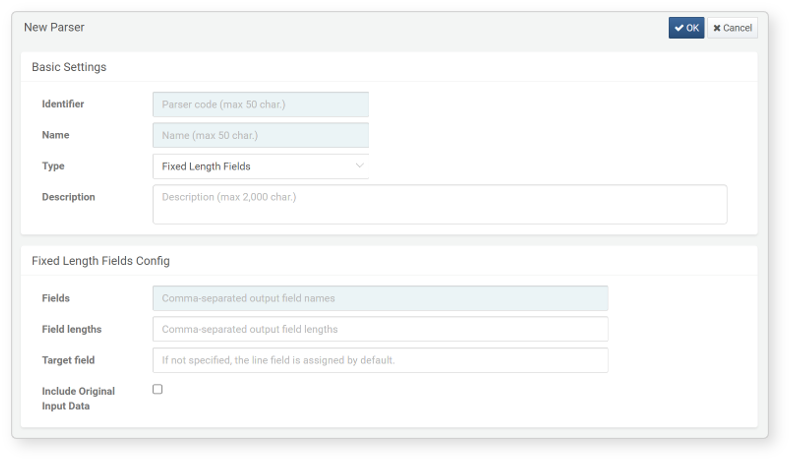
Whois

Whoisパーサーは、whoisコマンドの出力を処理します。このパーサーには追加で指定する属性はありません。



固定長フィールド

このパーサーは、固定長フィールドを持つデータを処理します。データは指定された長さ単位で切り出され、各フィールドに割り当てられ、順次解析されます。

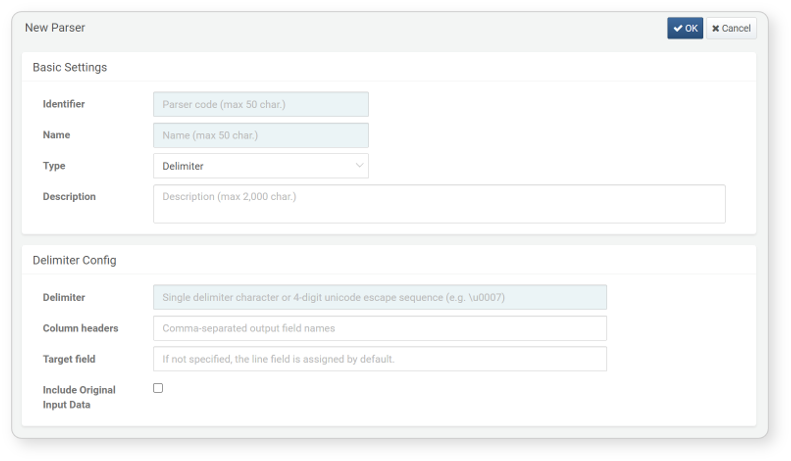


* **フィールド**：解析されたデータを割り当てるフィールド名のカンマ区切りリスト（デフォルト：なし、最大10,000文字）
* **フィールド長**：**フィールド名**ごとの文字数をカンマ区切りで指定（デフォルト：なし、最大10,000文字）
* **対象フィールド**：解析対象となる入力フィールド名（デフォルト：なし、最大255文字）。指定しない場合、**line**フィールドのデータが解析されます。
* **元の入力データを含める**：解析結果とともに元のフィールド値を含めるかどうか（デフォルト：未選択）。選択した場合、元データは**line**フィールドに含まれます。

固定長パーサーは、**フィールド長**に基づいてデータを解析します。定義された長さを超えるデータは無視されます。**フィールド長**リストが**フィールド**リストより短い場合、不足分のフィールドにはnullが割り当てられます。

区切り文字

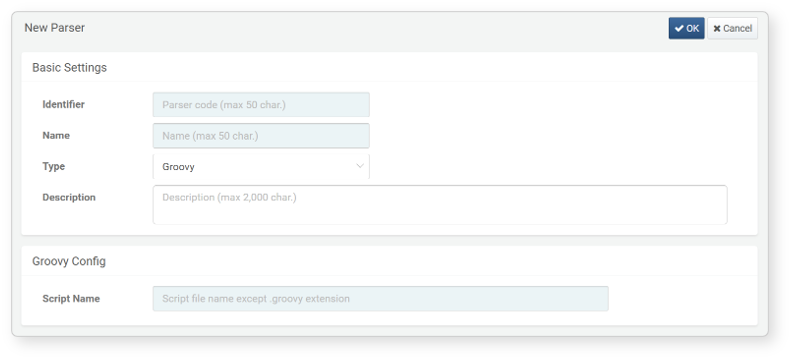
区切り文字パーサーは、指定された区切り文字で分割されたトークンを識別し、**カラムヘッダー**で指定されたフィールドに割り当ててデータを処理します。



* **区切り文字**：区切り文字として使用する文字（最小1文字、最大10文字）。\u002Cのような4桁のUnicodeエスケープシーケンスも使用可能です。Unicodeエスケープシーケンスは[SYMBL](https://symbl.cc/ja/unicode-table/)で検索できます。
* **カラムヘッダー**：解析されたデータを順番に割り当てるフィールド名のカンマ区切りリスト（デフォルト：なし、最大10,000文字）。指定しない場合、column0、column1、...、columnNがフィールド名として使用されます。
* **対象フィールド**：解析対象となる入力フィールド名（デフォルト：なし、最大255文字）。指定しない場合、**line**フィールドのデータが解析されます。
* **元の入力データを含める**：解析結果とともに元のフィールド値を含めるかどうか（デフォルト：未選択）。選択した場合、元データは**line**フィールドに含まれます。

Groovy

Groovyパーサーは、[Groovy](http://www.groovy-lang.org)言語で記述されたスクリプトを使用して入力データを処理します。



* **スクリプト名**：拡張子.groovyを除いたスクリプトファイル名（最大255文字）

Groovyは、PythonやRubyに着想を得た動的オブジェクト指向プログラミング言語で、Java Virtual Machine（JVM）上で動作します。スクリプトを実行するには、以下の制約を守る必要があります。

* スクリプトファイル名はCLASS\_NAME.groovyの形式である必要があります。
* Groovyパーサーは、ログプレッソ・ソナーのインストールディレクトリ内data/araqne-logdb-groovy/parser\_scriptsディレクトリに配置されたスクリプトファイルのみ実行できます。
* ログプレッソが提供するパッケージをインポートして使用する必要があります。必要に応じて以下のパッケージを利用してください。
* groovy.transform.CompileStatic（パフォーマンス向上のため推奨）
* org.araqne.logdb.groovy.GroovyQueryScript（必須）
* org.araqne.logdb.QueryStopReason
* org.araqne.logdb.Row（必須）
* org.araqne.logdb.RowBatch
* org.araqne.logdb.RowPipe

Groovyスクリプトのパフォーマンス向上のヒント

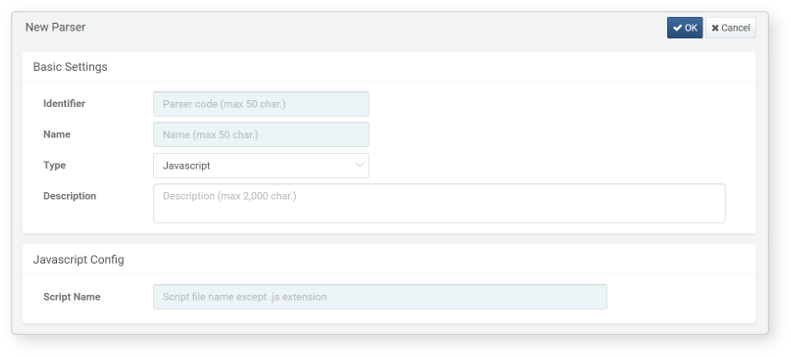
* 大量の文字列操作メソッドの使用を避ける：過度な文字列操作はJVMによるガベージコレクションを頻発させます。
* split()やtokenize()の代わりにindexOf()やsubstring()を使用する：コードは長くなりますが、処理性能が向上します。
* Pattern.compile()の繰り返し呼び出しを避ける。代わりにMatcher.reset()でMatcherインスタンスを再利用してください。
* 例外の使用を最小限に抑える：
* 頻繁な例外発生は処理速度を大幅に低下させます。
* 可能な限り条件分岐でエラーシナリオを処理してください。

以下は、data/araqne-logdb-groovy/parser\_scriptsディレクトリにToAscii.groovyとして保存されたサンプルスクリプトです。このスクリプトは、**payload**フィールド（PCAPファイルからデコードされたバイナリ値）から32～127の範囲の文字をASCII形式にエンコードします。

import groovy.transform.CompileStatic;import org.araqne.logdb.Row;import org.araqne.logdb.groovy.GroovyQueryScript;@CompileStatic(groovy.transform.TypeCheckingMode.SKIP)class ToAscii extends GroovyQueryScript {def void onRow(Row row) { byte[] payload = row.get('payload') char[] chars = new char[payload.length]; for (int i = 0; i < payload.length; i++) { char c = (char) payload[i] if (c < 32 || c > 126) c = '.' chars[i] = c } row.put('text', new String(chars)) pipe.onRow(row) }}

JavaScript

JavaScriptパーサーは、JavaScript言語で記述されたスクリプトを用いて入力データを処理します。



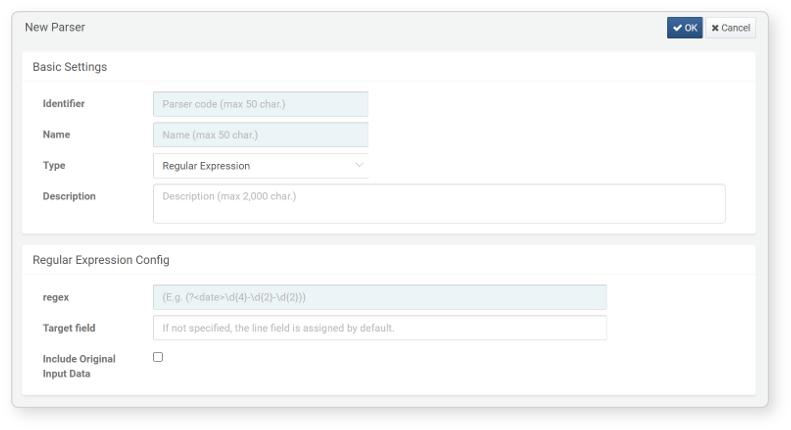
* **スクリプト名**: スクリプトファイル名（拡張子 .js を除く、最大255文字）。スクリプトファイルは、システムスイッチ araqne.log.data で指定されたディレクトリのサブディレクトリである araqne-logdb-nashorn/parser\_scripts ディレクトリ内に配置する必要があります。

以下は、data/araqne-logdb-nashorn/parser\_scripts ディレクトリに EaiTrParser.js という名前で保存されたサンプルスクリプトです。このスクリプトは **EUC-KR** エンコーディングで記録されたログファイルをパースするために設計されています。各フィールドは固定長であり、スクリプトはこれらの長さに基づいてフィールドに値を割り当てます。パース中にエラーが発生した場合は、エラー情報を error フィールドに格納します。

var stdout = java.lang.System.out;var jstring = Java.type("java.lang.String");function EaiTrParser() {}EaiTrParser.prototype.parse = function (m) { var b = m.body.getBytes("euc-kr"); try { m.a000\_if\_id = new jstring(b, 0, 16, "euc-kr").trim(); m.a001\_account = new jstring(b, 16, 11, "euc-kr").trim(); m.a002\_trd\_dt = new jstring(b, 27, 8, "euc-kr").trim(); m.a003\_trd\_sno = new jstring(b, 35, 10, "euc-kr").trim(); m.a004\_trd\_tp\_cd = new jstring(b, 45, 2, "euc-kr").trim(); m.a005\_act\_trd\_cfc\_cd = new jstring(b, 47, 2, "euc-kr").trim(); m.a006\_act\_trd\_dtl\_cd = new jstring(b, 49, 2, "euc-kr").trim(); m.a007\_act\_pdt\_llf\_cd = new jstring(b, 51, 2, "euc-kr").trim(); m.a008\_dca\_tp\_cd = new jstring(b, 53, 2, "euc-kr").trim(); m.a009\_sps\_cd = new jstring(b, 55, 5, "euc-kr").trim(); m.a010\_biz\_pcs\_no = new jstring(b, 60, 7, "euc-kr").trim(); m.a011\_act\_trd\_dit\_cd = new jstring(b, 67, 2, "euc-kr").trim(); m.a012\_can\_yn = new jstring(b, 69, 1, "euc-kr").trim(); m.a013\_org\_trd\_sno = new jstring(b, 70, 10, "euc-kr").trim(); m.a014\_rgd\_trd\_tp\_cd = new jstring(b, 80, 2, "euc-kr").trim(); m.a015\_dca\_rgd\_no = new jstring(b, 82, 7, "euc-kr").trim(); m.a016\_sct\_rgd\_no = new jstring(b, 89, 7, "euc-kr").trim(); m.a017\_iem\_cd = new jstring(b, 96, 12, "euc-kr").trim(); m.a018\_byn\_dt = new jstring(b, 108, 8, "euc-kr").trim(); m.a019\_syn\_ttn\_dit\_cd = new jstring(b, 116, 1, "euc-kr").trim(); m.a020\_trd\_qty = new jstring(b, 117, 18, "euc-kr").trim(); m.a021\_trd\_amt = new jstring(b, 135, 15, "euc-kr").trim(); m.a021\_trd\_amt\_dot = new jstring(b, 150, 3, "euc-kr").trim(); m.a022\_trd\_uit\_pr = new jstring(b, 153, 12, "euc-kr").trim(); m.a022\_trd\_uit\_pr\_dot = new jstring(b, 165, 3, "euc-kr").trim(); m.a023\_chk\_trd\_amt = new jstring(b, 168, 18, "euc-kr").trim(); m.a024\_ect\_chk\_trd\_amt = new jstring(b, 186, 18, "euc-kr").trim(); m.a025\_trd\_pna = new jstring(b, 204, 18, "euc-kr").trim(); m.a026\_int\_amt = new jstring(b, 222, 18, "euc-kr").trim(); m.a027\_trd\_dda = new jstring(b, 240, 18, "euc-kr").trim(); m.a028\_rit\_osm\_sel\_cst = new jstring(b, 258, 18, "euc-kr").trim(); m.a029\_trd\_orn\_fee = new jstring(b, 276, 12, "euc-kr").trim(); m.a029\_trd\_orn\_fee\_dot = new jstring(b, 288, 3, "euc-kr").trim(); m.a030\_trd\_tax = new jstring(b, 291, 18, "euc-kr").trim(); m.a031\_icm\_tax = new jstring(b, 309, 18, "euc-kr").trim(); m.a032\_rsd\_tax = new jstring(b, 327, 18, "euc-kr").trim(); m.a033\_fmfs\_tax = new jstring(b, 345, 18, "euc-kr").trim(); m.a034\_cyc\_icm\_tax = new jstring(b, 363, 18, "euc-kr").trim(); m.a035\_cyc\_rsd\_tax = new jstring(b, 381, 18, "euc-kr").trim(); m.a036\_frs\_fee = new jstring(b, 399, 18, "euc-kr").trim(); m.a037\_frs\_trd\_tax = new jstring(b, 417, 18, "euc-kr").trim(); m.a038\_frs\_icm\_tax = new jstring(b, 435, 18, "euc-kr").trim(); m.a039\_frs\_rsd\_tax = new jstring(b, 453, 18, "euc-kr").trim(); m.a040\_frs\_fmfs\_tax = new jstring(b, 471, 18, "euc-kr").trim(); m.a041\_fee\_rt = new jstring(b, 489, 6, "euc-kr").trim(); m.a041\_fee\_rt\_dot = new jstring(b, 495, 9, "euc-kr").trim(); m.a042\_ect\_txa = new jstring(b, 504, 18, "euc-kr").trim(); m.a043\_sby\_bse\_pr = new jstring(b, 522, 9, "euc-kr").trim(); m.a043\_sby\_bse\_pr\_dot = new jstring(b, 531, 6, "euc-kr").trim(); m.a044\_sas\_bse\_pr = new jstring(b, 537, 9, "euc-kr").trim(); m.a044\_sas\_bse\_pr\_dot = new jstring(b, 546, 6, "euc-kr").trim(); m.a045\_sas\_amt = new jstring(b, 552, 18, "euc-kr").trim(); m.a046\_tns\_amt = new jstring(b, 570, 18, "euc-kr").trim(); m.a047\_tns\_fee = new jstring(b, 588, 18, "euc-kr").trim(); m.a048\_trd\_bf\_dca = new jstring(b, 606, 18, "euc-kr").trim(); m.a049\_trd\_af\_dca = new jstring(b, 624, 18, "euc-kr").trim(); m.a050\_trd\_bf\_sba\_amt = new jstring(b, 642, 18, "euc-kr").trim(); m.a051\_trd\_af\_sba\_amt = new jstring(b, 660, 18, "euc-kr").trim(); m.a052\_trd\_bf\_bnc\_qty = new jstring(b, 678, 18, "euc-kr").trim(); m.a053\_trd\_af\_bnc\_qty = new jstring(b, 696, 18, "euc-kr").trim(); m.a054\_trd\_bf\_rba = new jstring(b, 714, 18, "euc-kr").trim(); m.a055\_trd\_af\_rba = new jstring(b, 732, 18, "euc-kr").trim(); m.a056\_rvb\_orn\_amt = new jstring(b, 750, 18, "euc-kr").trim(); m.a057\_rvb\_orn\_qty = new jstring(b, 768, 18, "euc-kr").trim(); m.a058\_cli\_pe\_fnm = new jstring(b, 786, 40, "euc-kr").trim(); m.a059\_abk\_uit\_pr = new jstring(b, 826, 12, "euc-kr").trim(); m.a060\_abk\_uit\_pr\_dot = new jstring(b, 838, 3, "euc-kr").trim(); m.a060\_ctc\_int\_rt = new jstring(b, 841, 3, "euc-kr").trim(); m.a061\_ctc\_int\_rt\_dot = new jstring(b, 844, 8, "euc-kr").trim(); m.a061\_opi\_ogt\_cfc\_cd = new jstring(b, 852, 4, "euc-kr").trim(); m.a062\_opi\_ogt\_cd = new jstring(b, 856, 12, "euc-kr").trim(); m.a063\_opi\_tab\_cd = new jstring(b, 868, 4, "euc-kr").trim(); m.a064\_ata\_opi\_act\_no = new jstring(b, 872, 20, "euc-kr").trim(); m.a065\_cfd\_trd\_yn = new jstring(b, 892, 1, "euc-kr").trim(); m.a066\_krx\_ntc\_trd\_cd = new jstring(b, 893, 5, "euc-kr").trim(); m.a067\_fcl\_sip\_no = new jstring(b, 898, 7, "euc-kr").trim(); m.a068\_cfd\_lon\_cd = new jstring(b, 905, 2, "euc-kr").trim(); m.a069\_lon\_dt = new jstring(b, 907, 8, "euc-kr").trim(); m.a070\_lon\_orn\_amt = new jstring(b, 915, 18, "euc-kr").trim(); m.a071\_lon\_rdp\_amt = new jstring(b, 933, 18, "euc-kr").trim(); m.a072\_lon\_int\_rt = new jstring(b, 951, 3, "euc-kr").trim(); m.a072\_lon\_int\_rt\_dt = new jstring(b, 954, 8, "euc-kr").trim(); m.a073\_xrn\_dt = new jstring(b, 962, 8, "euc-kr").trim(); m.a074\_lon\_int\_amt = new jstring(b, 970, 18, "euc-kr").trim(); m.a075\_trd\_bf\_lon\_amt = new jstring(b, 988, 18, "euc-kr").trim(); m.a076\_trd\_af\_lon\_amt = new jstring(b, 1006, 18, "euc-kr").trim(); m.a077\_trd\_bf\_lon\_qty = new jstring(b, 1024, 18, "euc-kr").trim(); m.a078\_trd\_af\_lon\_qty = new jstring(b, 1042, 18, "euc-kr").trim(); m.a079\_rdp\_dfr\_amt = new jstring(b, 1060, 18, "euc-kr").trim(); m.a080\_bf\_dd\_hhd\_chk\_use\_amt = new jstring(b, 1078, 18, "euc-kr").trim(); m.a081\_bf\_dd\_bnk\_chk\_use\_amt = new jstring(b, 1096, 18, "euc-kr").trim(); m.a082\_tdy\_hhd\_chk\_use\_amt = new jstring(b, 1114, 18, "euc-kr").trim(); m.a083\_tdy\_bnk\_chk\_use\_amt = new jstring(b, 1132, 18, "euc-kr").trim(); m.a084\_trd\_mdi\_cd = new jstring(b, 1150, 2, "euc-kr").trim(); m.a085\_rvb\_pbk\_trd\_yn = new jstring(b, 1152, 1, "euc-kr").trim(); m.a086\_sve\_trd\_yn = new jstring(b, 1153, 1, "euc-kr").trim(); m.a087\_fno\_trd\_yn = new jstring(b, 1154, 1, "euc-kr").trim(); m.a088\_ksfc\_pof\_stk\_trd\_yn = new jstring(b, 1155, 1, "euc-kr").trim(); m.a089\_fc\_trd\_yn = new jstring(b, 1156, 1, "euc-kr").trim(); m.a090\_dmt\_its\_trd\_yn = new jstring(b, 1157, 1, "euc-kr").trim(); m.a091\_ose\_its\_trd\_yn = new jstring(b, 1158, 1, "euc-kr").trim(); m.a092\_cd\_cp\_trd\_yn = new jstring(b, 1159, 1, "euc-kr").trim(); m.a093\_rp\_trd\_yn = new jstring(b, 1160, 1, "euc-kr").trim(); m.a094\_trs\_trd\_yn = new jstring(b, 1161, 1, "euc-kr").trim(); m.a095\_rcv\_trd\_yn = new jstring(b, 1162, 1, "euc-kr").trim(); m.a096\_bf\_lst\_mm\_pmt\_tal = new jstring(b, 1163, 18, "euc-kr").trim(); m.a097\_bf\_lst\_pmt\_dt = new jstring(b, 1181, 8, "euc-kr").trim(); m.a098\_rgs\_dt = new jstring(b, 1189, 8, "euc-kr").trim(); m.a099\_rgs\_tm = new jstring(b, 1197, 8, "euc-kr").trim(); m.a100\_rgs\_tab\_cd = new jstring(b, 1205, 4, "euc-kr").trim(); m.a101\_rgs\_emp\_no = new jstring(b, 1209, 6, "euc-kr").trim(); m.a102\_rgs\_trm\_no = new jstring(b, 1215, 8, "euc-kr").trim(); m.a103\_rgs\_cuc\_mdi\_cd = new jstring(b, 1223, 2, "euc-kr").trim(); m.a104\_rgs\_usr\_ip\_adr = new jstring(b, 1225, 32, "euc-kr").trim(); m.a105\_rgs\_hts\_usr\_id = new jstring(b, 1257, 8, "euc-kr").trim(); m.a106\_alt\_dt = new jstring(b, 1265, 8, "euc-kr").trim(); m.a107\_alt\_tm = new jstring(b, 1273, 8, "euc-kr").trim(); m.a108\_alt\_tab\_cd = new jstring(b, 1281, 4, "euc-kr").trim(); m.a109\_alt\_emp\_no = new jstring(b, 1285, 6, "euc-kr").trim(); m.a110\_alt\_trm\_no = new jstring(b, 1291, 8, "euc-kr").trim(); m.a111\_alt\_cuc\_mdi\_cd = new jstring(b, 1299, 2, "euc-kr").trim(); m.a112\_alt\_usr\_ip\_adr = new jstring(b, 1301, 32, "euc-kr").trim(); m.a113\_alt\_hts\_usr\_id = new jstring(b, 1333, 8, "euc-kr").trim(); m.a114\_amn\_tab\_cd = new jstring(b, 1341, 4, "euc-kr").trim(); m.a115\_amn\_mo\_tab\_cd = new jstring(b, 1345, 4, "euc-kr").trim(); m.filler = new jstring(b, 1349, 651, "euc-kr").trim(); } catch (error) { m.error = error; } return m;};

正規表現

正規表現パーサーは、指定されたパターンを用いて入力データからフィールド値を抽出します。



* **正規表現**: フィールド値を抽出するためのパターン（最大10,000文字）。マッチした値をフィールドに割り当てるには、式内でフィールド名を名前付きキャプチャグループとして指定する必要があります（例：(?<date>\d{4}-\d{2}-\d{2})、ここで date がマッチした値を受け取るフィールド名です）。
* **対象フィールド**: パース対象となる入力フィールド名（最大255文字）。未指定の場合、**line** フィールドのデータがパースされます。
* **元の入力データを含める**: パース結果とともに元のフィールド値を含めるかどうか（デフォルト：未選択）。選択した場合、元データが **line** フィールドに含まれます。

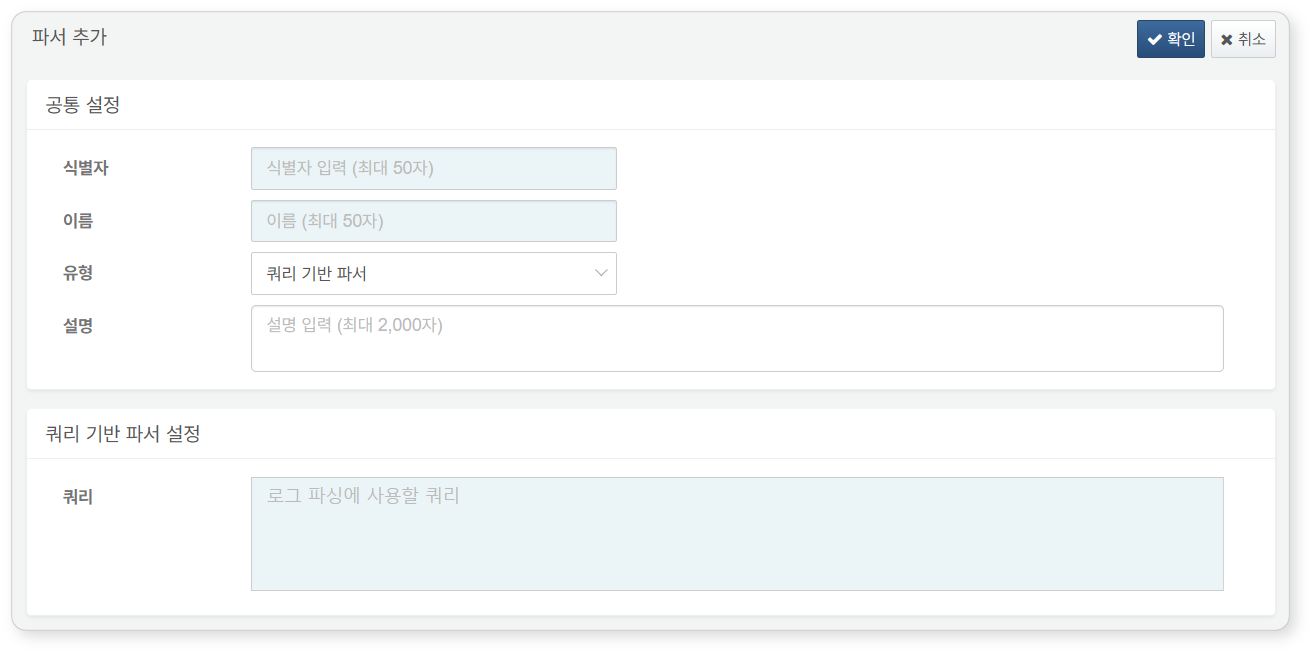
正規表現は柔軟かつ強力な文字列マッチング機能を提供しますが、実装が複雑になる場合があります。このパーサーを効果的に利用するには、正規表現の仕組みを十分に理解しておくことが重要です。

以下の正規表現は、Webサーバーログをパースします。

この正規表現は、ApacheおよびNginx Webサーバーのデフォルトログフォーマットを前提としています。ご利用のログフォーマットが変更されている場合は、正規表現も適宜修正してください。

クエリベースパーサー

クエリベースパーサーは、[ログプレッソ・クエリ](https://docs.logpresso.comnull)を利用して入力データを処理します。クエリ構文を活用することで、他のパーサーよりも高度かつ柔軟なパースが可能です。ユーザー定義パーサーを作成する際のデフォルトパーサータイプであり、最も一般的に使用されるパーサーです。



* **クエリ文字列**：入力データをパースするためのログプレッソ・クエリ文（最大50,000文字）。

以下は、Webサーバーログから抽出したユーザーエージェント情報を分類するクエリ例です。

eval agent\_type = case( user\_agent == "\*Googlebot\*", "google", user\_agent == "\*GoogleDocs\*", "google docs", user\_agent == "\*bingbot\*" or user\_agent == "\*BingPreview\*", "bing", user\_agent == "\*facebookexternalhit\*", "facebook", user\_agent == "\*Yandex\*", "yandex", user\_agent == "\*api.slack.com\*", "slack", user\_agent == "\*CensysInspect\*", "censys", user\_agent == "\*Chrome\*", "chrome", user\_agent == "\*Safari\*", "safari", user\_agent == "\*Firefox\*", "firefox", user\_agent == "\*Python\*" or user\_agent == "\*python\*", "python", user\_agent == "\*Go-http-client\*", "golang", user\_agent == "\*MJ12bot\*", "mj12", user\_agent == "\*Expanse, a Palo Alto Networks\*", "xpanse", user\_agent == "\*SemrushBot\*", "semrush", user\_agent == "\*PetalBot\*", "petal", user\_agent == "\*AhrefsBot\*", "ahrefs", "misc" )

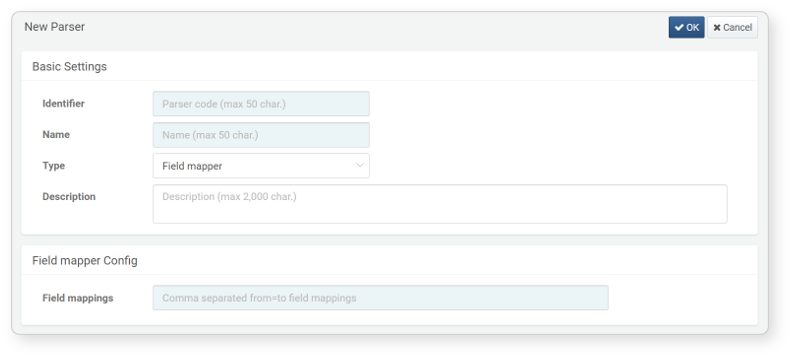
クエリベースパーサーでは、入力データがすでに利用可能な状態で処理されるため、クエリは通常、データを[変換](https://docs.logpresso.comnull)したり、他のデータセットと[マッピング](https://docs.logpresso.comnull)や[マージ](https://docs.logpresso.comnull)するコマンドから始まります。

廃止予定パーサー

以下のパーサーは後方互換性のために提供されていますが、廃止予定です。代替として[クエリベースパーサー](https://docs.logpresso.comnull)の利用を推奨します。

フィールドマッパー

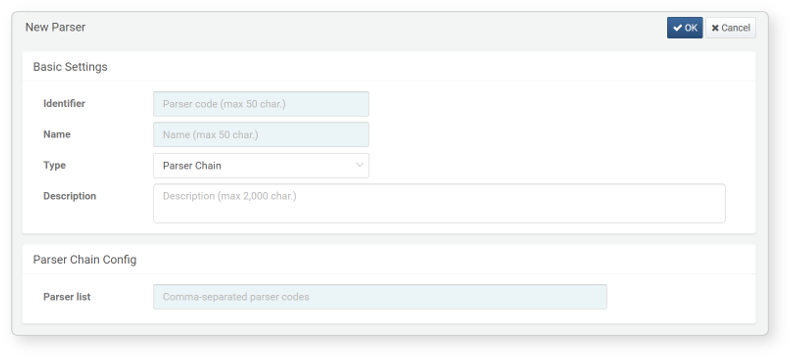
入力データ内の特定フィールド名をリネームします。



* **フィールドマッピング**：**元のフィールド名=新しいフィールド名**の形式で、カンマ（,）区切りで入力します（最大10,000文字）。

パーサーチェーン

複数のパーサーを順次適用して入力データを処理します。チェーン内の各パーサーは、前のパーサーの出力を入力として受け取ります。

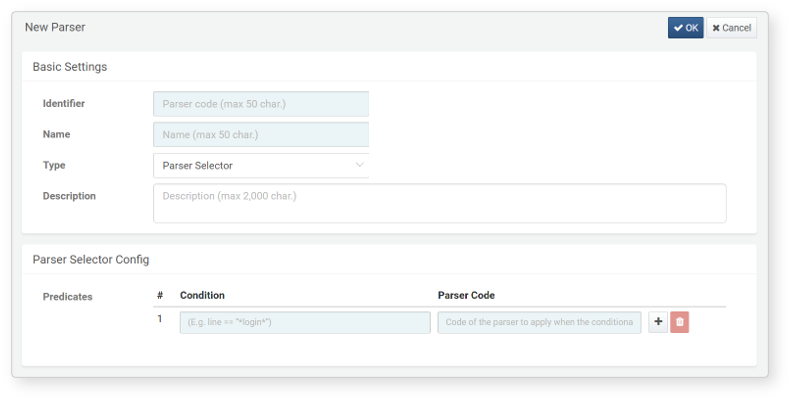


* **パーサーリスト**：適用順にカンマ区切りで記述したパーサー識別子のリスト（最大1,000文字）。

パーサーチェーンが長くなりすぎると、ログ分析のパフォーマンスが低下する可能性があります。必要なパーサーのみを使用してください。

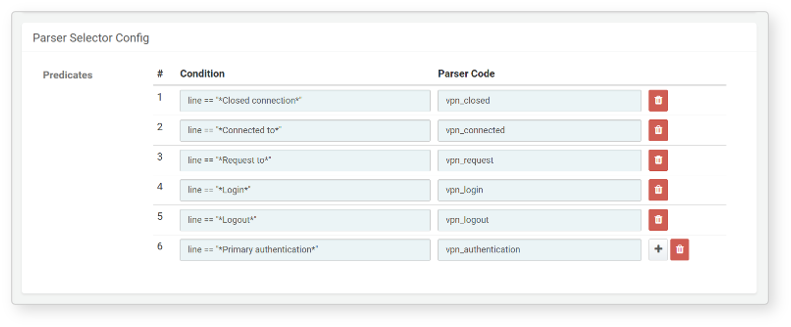
パーサーセレクター

指定した条件に一致する最初のパーサーを選択し、入力データを処理します。



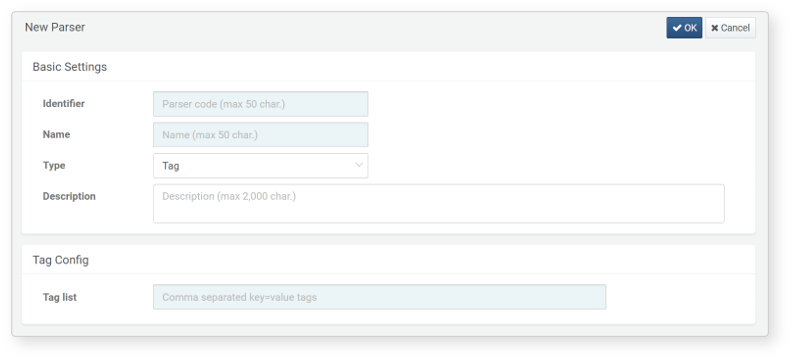
* **述語**：クエリ式で記述された条件で、ブール値を返します。式がtrueの場合、対応するパーサー識別子が適用されます。
* **条件**：ブール値（trueまたはfalse、最大1,000文字）を返すクエリ条件。1つの入力レコードに複数の条件が一致した場合、最初に一致した条件に対応するパーサーが選択されます。
* **パーサーコード**：条件が成立した場合に適用されるパーサーの識別子（最大50文字）。

パーサーセレクターは、入力レコードに対して条件式を評価し、最初にtrueとなった条件に紐づくパーサーを適用します。条件式は重複しないように記述し、すべての入力ケースを網羅することを推奨します。



タグ

タグパーサーは、入力データのメタデータにキーと値のタグを付与してパースします。

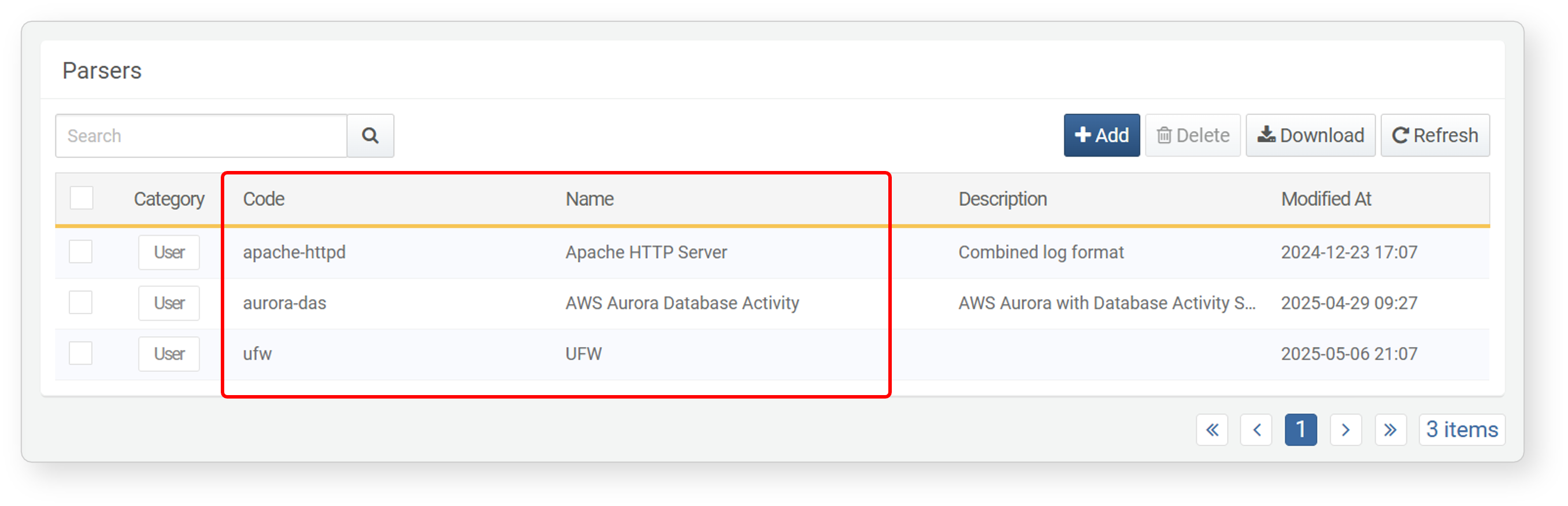


* **タグリスト**：key=value形式のメタデータタグをカンマ区切りで入力（最大10,000文字）

#### パーサーの編集

パーサーを編集するには、以下の手順に従います。

編集したいパーサーの**コード（識別子）\*\*または\*\*名称**をリストからクリックします。



**パーサー編集**画面で、必要な情報を修正し、**OK**をクリックします。

* 編集可能なプロパティの詳細は[パーサー追加](https://docs.logpresso.comnull)を参照してください。
* **識別子**フィールドは編集できません。変更する場合は、パーサーを削除して再作成してください。

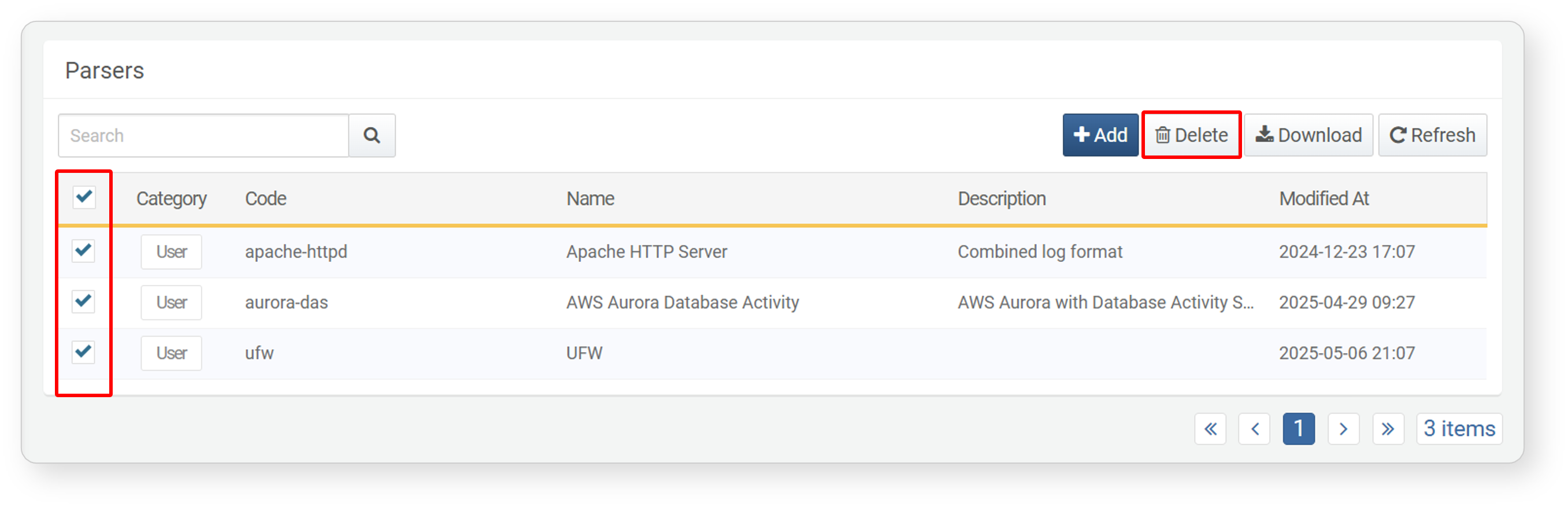
アプリパーサーを編集または変更しないでください。アプリの動作に支障をきたす恐れがあります。

#### パーサーの削除

パーサーを削除するには、以下の手順に従います。

削除したいパーサー情報の行のチェックボックスを選択します。

ツールバーの**削除**をクリックします。



**パーサー削除**ダイアログで、削除対象のパーサーリストを確認し、**削除**をクリックします。削除しない場合は**キャンセル**をクリックしてください。

パーサーの削除に失敗した場合は、**パーサー削除失敗**ダイアログで失敗理由を確認できます。

アプリパーサーを削除しないでください。アプリの動作に支障をきたす恐れがあります。