### Parser

#### Overview

A parser is used to identify input data on a record-by-record basis and to split the identified records into fields.



Parsers have the following characteristics:

**Preservation of Original Records**

Parsers are configured to preserve the original records. Instead of processing the original data, they identify and extract fields from the original to create separate fields. In Logpresso queries, the original record is typically output in the **line** field.

**Dedicated Parsers for Log Formats**

Even for the same type of system, logs are formatted differently depending on the manufacturer. For example, Palo Alto Networks firewall logs are in CSV format, while Fortinet FortiGate firewall logs use a key=value format with whitespace as the delimiter. Logpresso Sonar uses different parsers to extract field values from the original logs according to each log format.

Below is an example of an original record taken from a Palo Alto Networks firewall.

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,

To check the parsed result of the above log, install the [Palo Alto Networks Firewall](https://logpresso.store/ko/apps/paloalto-ngfw) app from the Logpresso Store, then execute the following query in **Analysis > Query**.

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

Below is an example of an original record taken from a Fortinet FortiGate firewall.

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"

To check the parsed result of the above log, install the [FortiGate](https://logpresso.store/ko/apps/fortigate) app from the Logpresso Store, then execute the following query in **Analysis > Query**.

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

As demonstrated, parsers perform the task of transforming data in different formats into a normalized form.

App Parsers and Basic Parsers

Parsers can be categorized into basic parsers and app parsers.

**App Parsers**

Most parsers are provided along with the installation of an [app](https://docs.logpresso.comnull). App parsers come pre-configured with all the necessary settings to extract fields from data collected from the integrated system, so there are rarely options that need to be configured separately.

Do not modify or delete app parsers. This may cause issues with the operation of the app.

**Basic Parsers**

These are parsers that can be defined and used by the cluster administrator when no apps are installed. After installing Logpresso Sonar, if you check the [list of parsers](https://docs.logpresso.comnull), you will see that no parsers are available. If additional parsers are needed for purposes other than app parsers, the cluster administrator can add them for use.

Execution Location

The execution location of parsers varies depending on the form of the Logpresso Sonar [cluster](https://docs.logpresso.comnull). As shown in the table below, parsers execute on the nodes where data is stored in tables.

|  |  |  |
| --- | --- | --- |
| Cluster | Configuration Nodes (Each node supports redundancy) | Parser Execution Node |
| Tier 1 | Analysis Node | Analysis Node |
| Tier 2 | Collection Node -> Analysis Node | Collection Node |
| Tier 3 | Forwarding Node -> Collection Node -> Analysis Node | Collection Node |

#### Viewing/Search Parser List

You can view or search the parser list in **Collection > Parser**. The following image shows the parser list that can be seen after installing the [Palo Alto Networks Firewall](https://logpresso.store/ko/apps/paloalto-ngfw), [BlueMax NGF](https://logpresso.store/ko/apps/bluemax-ngf), and [AhnLab TrusGuard](https://logpresso.store/ko/apps/ahnlab-trusguard) apps.



* **Identifier**: The parser identifier used in query statements
* **Name**: The name used to identify the parser
* **Description**: Additional information about the parser
* **Modification Date**: The date the app parser was created or last modified

To find a specific parser in the parser list, use the search tool in the toolbar. The search tool will find and display parsers that contain the entered words in **Identifier**, **Name**, or **Description**. The parser search tool is case-insensitive.



#### Adding a Parser

In most operational environments, using app parsers is sufficient. However, if there is no method to parse logs collected from the data source, or if the functionality of the parsers provided by the app is not suitable for the operational environment, you can add a parser for use.

To add a new parser:

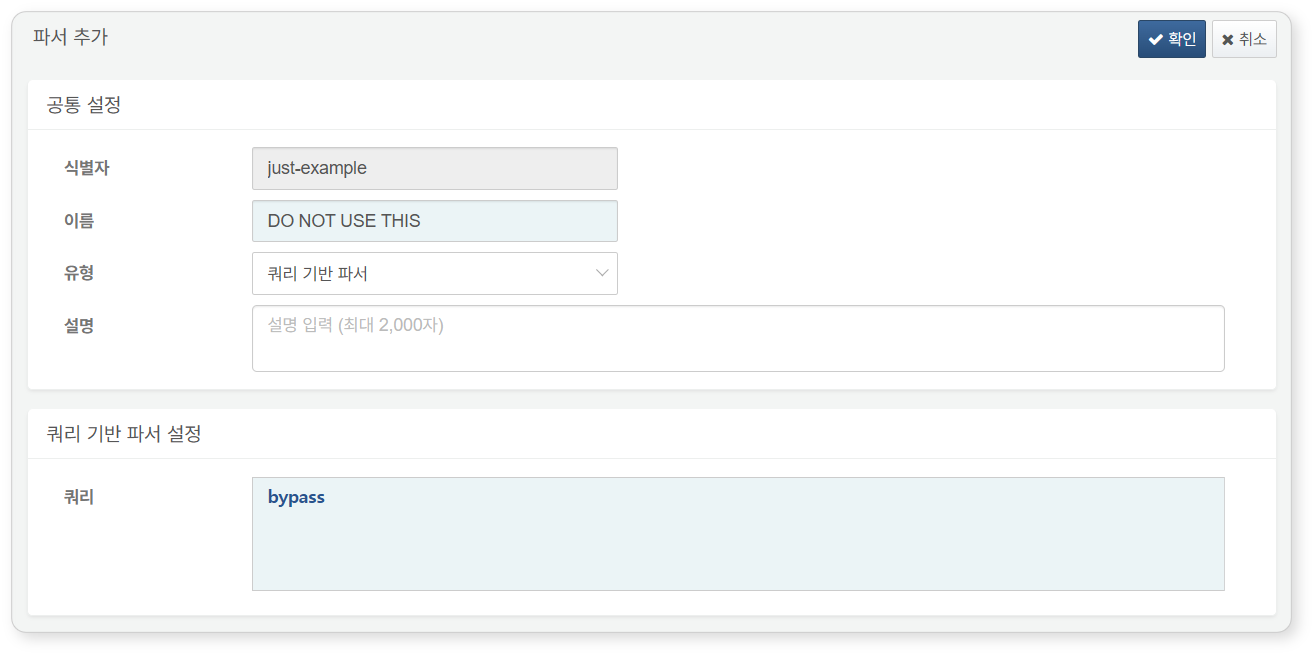
Click **Add** in the toolbar under **Collection > Parser**.



Refer to the [Common Settings](https://docs.logpresso.comnull) and [Basic Parser Settings](https://docs.logpresso.comnull) to enter or select the necessary values for parser configuration, then click **OK** in the upper right corner.

Common Settings

In the **Common Settings**, enter or select the parser's **Identifier**, **Name**, **Type**, and **Description**. Depending on the type, there may be additional properties to configure.



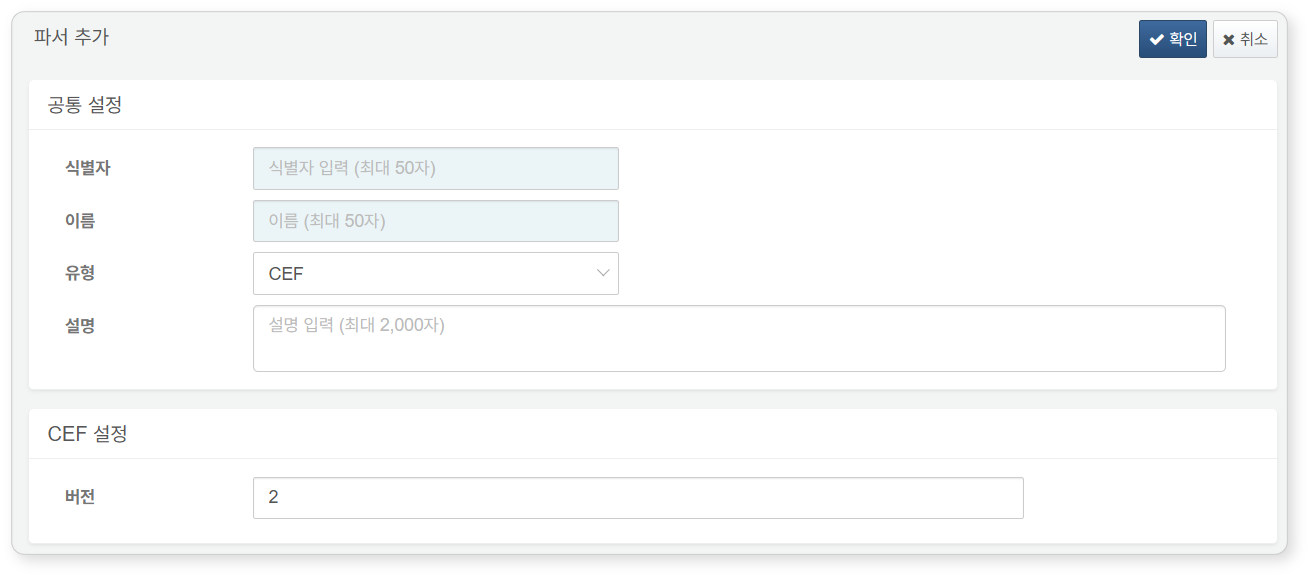
* **Identifier**: The parser identifier used in query statements. It must be a unique name that does not overlap with other parser identifiers.
* **Name**: The name assigned for users to identify the parser in the web console.
* **Type**: The type of input data or the method to apply for parsing the input data (default: [Query-Based Parser](https://docs.logpresso.comnull)).
* For properties that need to be set according to the type, refer to the [Basic Parser Settings](https://docs.logpresso.comnull). There may be no properties to enter depending on the parser type.
* **Description**: A description of the parser. Use this field to provide additional information about the parser.

Basic Parser Settings

This section describes the properties for each type of [Basic Parser](https://docs.logpresso.comnull). For app parsers, refer to the app documentation in the Logpresso Store.

CEF

Parses input data in Common Event Format (CEF). CEF is a standard used when integrating with ESM or SIEM systems like ArcSight.

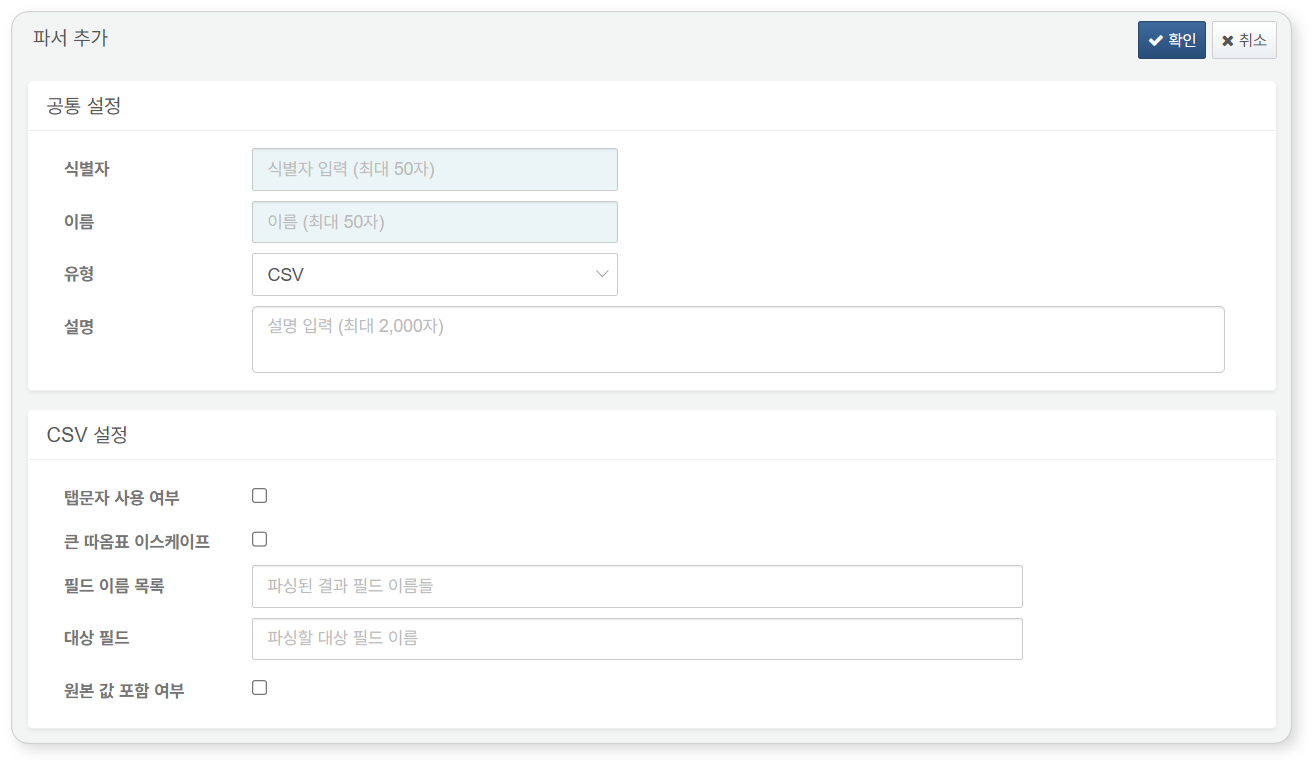


* **Version**: Parser version (default: 2). Version 1 is for backward compatibility. Use version 2 to define a new CEF parser.
* **1**: Parses according to CEF specification version 23 (reference: [CEF Log Format](https://ko.logpresso.com/ko/blog/2017-03-25-cef-log-format))
* **2**: Parses according to CEF specification version 25. Supports field name normalization as shown in the following table.

|  |  |  |
| --- | --- | --- |
| Before Normalization | After Normalization | Type |
| src | src\_ip | IP Address |
| spt | src\_port | 32-bit Integer |
| dst | dst\_ip | IP Address |
| dpt | dst\_port | 32-bit Integer |
| sourceTranslatedAddress | nat\_src\_ip | IP Address |
| sourceTranslatedPort | nat\_src\_port | 32-bit Integer |
| destinationTranslatedAddress | nat\_dst\_ip | IP Address |
| destinationTranslatedPort | nat\_dst\_port | 32-bit Integer |
| dvc | device\_ip | IP Address |
| dvchost | device\_host | String |
| dvcmac | device\_mac | String |
| dvcpid | device\_pid | 64-bit Integer |
| sourceDnsDomain | src\_domain | String |
| sourceServiceName | src\_service | String |
| destinationDnsDomain | dst\_domain | String |
| destinationServiceName | dst\_service | String |
| shost | src\_host | String |
| smac | src\_mac | String |
| dhost | dst\_host | String |
| dmac | dst\_mac | String |
| suid | src\_user\_id | String |
| suser | src\_user | String |
| duid | dst\_user\_id | String |
| duser | dst\_user | String |
| spid | src\_pid | 64-bit Integer |
| sproc | src\_process | String |
| dpid | dst\_pid | 64-bit Integer |
| dproc | dst\_process | String |
| fileCreateTime | file\_ctime | Date/Time |
| fileModificationTime | file\_mtime | Date/Time |
| fname | file\_name | String |
| filePath | file\_path | String |
| fileType | file\_type | String |
| fsize | file\_size | 64-bit Integer |
| fileHash | file\_hash | String |
| in | rcvd\_bytes | 64-bit Integer |
| out | sent\_bytes | 64-bit Integer |
| proto | protocol | String |
| app | app | String |
| act | action | String |
| cat | category | String |
| cnt | event\_count | 32-bit Integer |

CSV

Receives input data in CSV (Comma-Separated Values) or TSV (Tab-Separated Values) format. It parses by matching each token identified by the delimiter (comma or tab character) against the **Field Name List** and assigning them to fields in order.

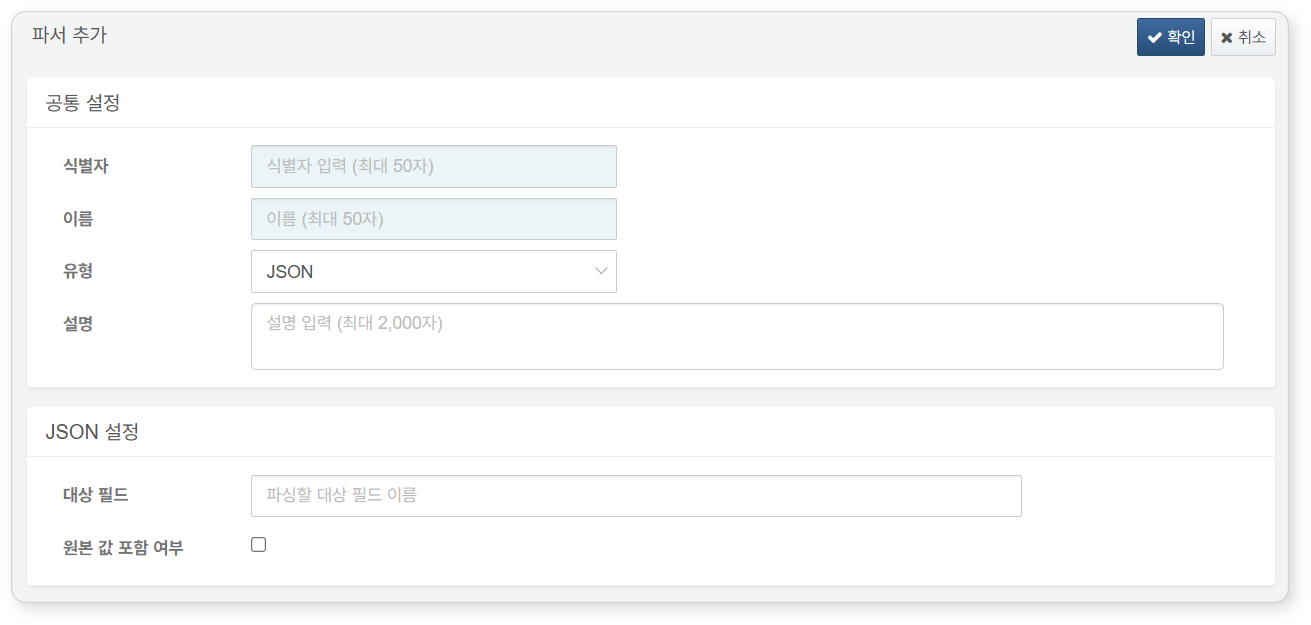


* **Use Tab Character**: Whether to use a tab character instead of a comma (default: not selected)
* **Escape Double Quotes**: Whether to use double quotes as an escape character for special characters (default: not selected)
* **Field Name List**: Enter the names of the parsed result fields separated by commas (default: none, maximum 10,000 characters). If not specified, tokens will be assigned to column0, column1, ... columnN fields.
* **Target Field**: The name of the input field to parse (default: none, maximum 500 characters). If not specified, it parses the data in the **line** field.
* **Include Original Value**: Whether to include the original field value in addition to the parsed CSV result (default: not selected). The original data will be included in the **line** field.

The CSV parser operates by assigning each token separated by a comma (,) or tab character in order to column0, column1, ... columnN fields. During this process, the names of column0, column1, ... columnN fields are changed to those specified in the **Field Name List**. If the **Field Name List** specifies 10 field names and there are 13 tokens, the 11th to 13th tokens will be assigned to column10, column11, and column12, respectively. Conversely, if there are not enough tokens, null will be assigned to those fields.

JSON

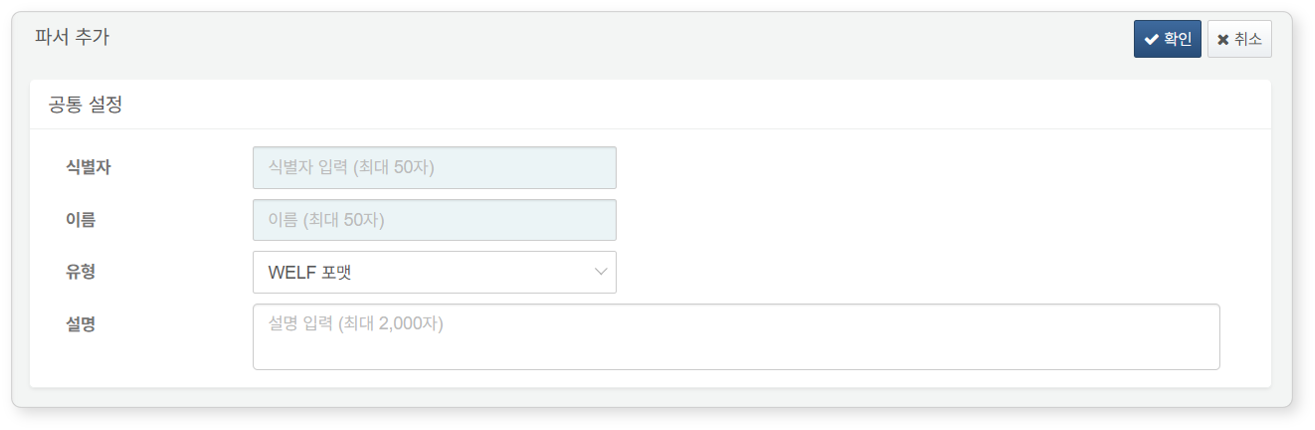
Parses input data in JSON format. If the input data is nested JSON, the JSON data within the JSON data will be assigned to composite fields structured as maps or arrays.



* **Target Field**: The name of the input field to parse (default: none, maximum 255 characters). If not specified, it parses the data in the **line** field.
* **Include Original Value**: Whether to include the original field value in addition to the parsed JSON result (default: not selected). The original data will be included in the **line** field.

WELF

Parses input data in WELF (WebTrends Enhanced Log Format). The WELF parser does not have any properties to specify.



Whois

Parses the output of the whois command. The Whois parser does not have any properties to