

# EBNF rule explanation

Perfect 👍 — let's make it **explicit long-form** so there's no doubt that "TEMPLATE" and `recipientAddressSource` are mutually exclusive.

Here's the revised grammar:

```
ebnf

mergeMultiDocWithTemplateParams =
  documentsToMerge
  + ( [ recipientAddressSource ]
    | [ "TEMPLATE" ] )
  + jobTemplate
  + paymentDetails
  + [ tags ] ;
```

## Why this is explicit

- `documentsToMerge` → always required.
- The `( ... | ... )` means you must choose **either**:
  - `[ recipientAddressSource ]` → zero or one recipient address, **or**
  - `[ "TEMPLATE" ]` → zero or one "TEMPLATE".
- You can't have both at once.
- `jobTemplate` and `paymentDetails` remain required.
- `tags` stays optional.

### ✓ Allowed examples:

- **No recipient:** just `documentsToMerge`, `jobTemplate`, `paymentDetails`.
- **With recipient:** `documentsToMerge`, `recipientAddressSource`, `jobTemplate`, `paymentDetails`.
- **With TEMPLATE:** `documentsToMerge`, "TEMPLATE", `jobTemplate`, `paymentDetails`.

### ✗ Not allowed: both "TEMPLATE" and `recipientAddressSource`.

Do you also want me to add a **comment block with these 3 explicit shapes** (like mini-examples under the grammar), so it's crystal clear for developers using the spec?



yes

Great 👍 — here's the **expanded grammar plus inline documentation** with explicit shapes listed.

```
ebnf

(*
Use Case 3: I want to merge multiple documents, send them to a single recipient, and use a job
template.
Endpoint: POST /jobs/multi-doc-merge-job-template

Valid Shapes:
1. With no recipient:
    documentsToMerge + jobTemplate + paymentDetails + [ tags ]

2. With recipientAddressSource:
    documentsToMerge + recipientAddressSource + jobTemplate + paymentDetails + [ tags ]

3. With TEMPLATE (instead of recipientAddressSource):
    documentsToMerge + "TEMPLATE" + jobTemplate + paymentDetails + [ tags ]

Note:
- You cannot provide both "TEMPLATE" and recipientAddressSource at the same time.
- jobTemplate and paymentDetails are always required.
- tags is always optional.
*)
mergeMultiDocWithTemplateParams =
    documentsToMerge
    + ( [ recipientAddressSource ]
      | [ "TEMPLATE" ] )
    + jobTemplate
    + paymentDetails
    + [ tags ] ;
```

This way, anyone reading the rule will see both the **formal grammar** and the **developer-friendly examples** in the comment block.

Would you like me to do the same treatment for your **single-doc case** (Use Case 1), so both specs follow the exact same “grammar + explicit shapes” format?



yes

Perfect — here’s the **single-doc case (Use Case 1)** rewritten with the same “grammar + explicit shapes” style we just applied to your multi-doc case.

```
ebnf

(*
Use Case 1: I want to submit a single document to one or more recipients using a job template.
Endpoint: POST /jobs/single-doc-job-template

Valid Shapes:
1. With document only:
    documentSourceIdentifier + jobTemplate + paymentDetails + [ tags ]

2. With recipients only:
    recipientAddressSource + jobTemplate + paymentDetails + [ tags ]

3. With both document and recipients:
    documentSourceIdentifier + recipientAddressSource + jobTemplate + paymentDetails + [
tags ]

4. With TEMPLATE instead of document:
    "TEMPLATE" + recipientAddressSource + jobTemplate + paymentDetails + [ tags ]

5. With TEMPLATE instead of recipients:
```

```
documentSourceIdentifier + "TEMPLATE" + jobTemplate + paymentDetails + [ tags ]
```

Notes:

- "TEMPLATE" may replace either the documentSourceIdentifier or the recipientAddressSource, but never both.
- jobTemplate and paymentDetails are always required.
- tags is always optional.

\*)

```
submitSingleDocWithTemplateParams =
  ( ( "TEMPLATE" | documentSourceIdentifier ) + { recipientAddressSource }
    | ( "TEMPLATE" | { recipientAddressSource } ) + documentSourceIdentifier
    | ( documentSourceIdentifier + { recipientAddressSource } ) )
  + jobTemplate
  + paymentDetails
  + [ tags ] ;
```

✓ Now both **single-doc** and **multi-doc** rules follow the same pattern:

- Formal grammar.
- Explicit shapes in a comment block.
- Clear notes on "TEMPLATE" usage.

Would you like me to also prepare a **summary section** that lists all your use cases (1, 2, 3, etc.) side by side in this same format, so your spec has a consistent developer-facing reference?