要件がはっきりしているので、既存の設計に合わせて **「エンコーディング別に JSON を扱うサブルーチン」** を追加する形で実装例を提示します。
（RHEL9 / Perl 5.32 / CGI / Template Toolkit 前提）

ポイントは以下です：

- JSON パース：JSON または JSON::XS を使用
- エンコーディング：Encode を使用（UTF-8 ⇔ Shift_JIS）
- CGI での取得：POST body を直接読む（$cgi->param は JSON に不向きな場合あり）

☑ 事前に必要なモジュール

```
use strict;
use warnings;
use CGI;
use JSON;
use Encode;
```

① UTF-8 JSON リクエスト取得

```
sub get_utf8_json_request {
    my $cgi = CGI->new;

    # POST body 取得
    my $raw = $cgi->param('POSTDATA');
    if (!defined $raw) {
        read(STDIN, $raw, $ENV{'CONTENT_LENGTH'});
    }
```

```
9
10      # UTF-8 として decode（念のため）
11      my $utf8_str = Encode::decode('UTF-8', $raw);
12
13      # JSON を Perl データ構造へ
14      my $json = JSON->new->utf8(1);
15      my $data = eval { $json->decode($utf8_str) };
16
17      if ($@) {
18          die "JSON parse error (UTF-8): $@";
19      }
20
21      return $data;
22  }
```

② Shift_JIS JSON リクエスト取得

```
1   sub get_sjis_json_request {
2       my $cgi = CGI->new;
3
4       my $raw = $cgi->param('POSTDATA');
5       if (!defined $raw) {
6           read(STDIN, $raw, $ENV{'CONTENT_LENGTH'});
7       }
8
9       # Shift_JIS → UTF-8 へ変換
10      my $utf8_str = Encode::decode('Shift_JIS', $raw);
11
12      my $json = JSON->new->utf8(1);
```

```
    my $data = eval { $json->decode($utf8_str) };

    if ($@) {
        die "JSON parse error (SJIS): $@";
    }

    return $data;
}
```

③ UTF-8 JSON レスポンス

```
sub output_utf8_json {
    my ($data) = @_;

    my $json = JSON->new->utf8(1)->pretty(0);
    my $json_text = $json->encode($data);

    print "Content-Type: application/json; charset=UTF-8\n\n";
    print $json_text;
}
```

④ Shift_JIS JSON レスポンス

```
sub output_sjis_json {
    my ($data) = @_;

    my $json = JSON->new->utf8(1)->pretty(0);

```

```
    # JSON 生成（UTF-8）
    my $utf8_json = $json->encode($data);

    # UTF-8 → Shift_JIS へ変換
     my $sjis_json = Encode::encode('Shift_JIS', $utf8_json);

     print "Content-Type: application/json;
charset=Shift_JIS\n\n";
     print $sjis_json;
}
```

☑ 補足（重要）

1. CGI->param('POSTDATA')について

- application/json の場合、通常の param では取れないため
- read(STDIN...) フォールバックありで OK

---

2. JSON は基本 UTF-8 が標準

- RFC 的には UTF-8 が基本
- Shift_JIS JSON は「非標準運用」

👉 可能なら UTF-8 に統一推奨

---

3. エラー処理

本番では die ではなく：

```
  return undef;
```

や

```
warn "JSON parse error: $@";
```

が望ましい

---

4. Module として統合例

```
package MiscCGIUtil;

sub get_utf8_json_request {...}
sub get_sjis_json_request {...}
sub output_utf8_json {...}
sub output_sjis_json {...}

1;
```

☑ 使い方例

```
my $req = MiscCGIUtil::get_utf8_json_request();

my $res = {
    status => 'OK',
    data   => $req,
};

MiscCGIUtil::output_utf8_json($res);
```

☑ まとめ

今回追加したもの：

処理	関数
UTF-8 JSON 受信	get_utf8_json_request
SJIS JSON 受信	get_sjis_json_request
UTF-8 JSON 送信	output_utf8_json
SJIS JSON 送信	output_sjis_json

必要なら：

- CGI.pm を使わない高速版（Plack/PSGI）
- File::Slurp と組み合わせた JSON 保存
- Template Toolkit 内で JSON 埋め込み

なども設計できます 👍