Guia: Como Criar JSONs de Configuração do Simulador

Este documento fornece uma referência completa para a criação dos arquivos de configuração JSON utilizados pelo Simulador Avançado TTN. Compreender estas estruturas é essencial para definir cenários de simulação e formatos de payload personalizados.

1. Visão Geral: O Sistema de Dois Ficheiros

O simulador utiliza um sistema de configuração de dois níveis para máxima flexibilidade:

1. `simulation_config.json` **(As configurações da simulação):** Este é o ficheiro principal que carrega. Define os parâmetros de alto nível da simulação: *que* tipo de simulação executar (ex: `application-uplink`), o seu alvo (ex: que aplicação/dispositivo), e quaisquer condições especiais (flags, envio periódico).
2. `payload_template.json` **(A forma do payload):** Este ficheiro é referenciado pela configuração da simulação. Fornece um projeto detalhado e de baixo nível sobre *como* construir um payload binário, campo a campo. Esta separação permite reutilizar definições de payload complexas em muitos cenários de simulação diferentes.

2. O Arquivo de Configuração da Simulação (`simulation_config.json`)

Este arquivo orquestra toda a simulação.

Exemplo:

```
{
  "simulation_type": "application-uplink",
  "description": "Uma simulação periódica que envia dados de sensor.",
  "periodic_settings": {
    "interval": 15,
    "enabled_on_load": false
  },
  "common_args": ["minha-app-de-teste", "meu-dispositivo-de-teste"],
  "flags": {
    "f-port": 2,
    "confirmed": true
  },
  "payload_source": {
```

```
    "type": "json_template",
    "file": "complex_payload.json"
  }
}
```

Campos de Nível Superior

Chave	Tipo	Obrigatório?	Descrição
`simulation_type`	String	**Sim**	O subcomando `ttn-lw-cli simulate` a ser executado. Valores comuns: `application-uplink`, `lorawan-uplink`, `gateway-status`. Consulte a Documentação Oficial do `simulate` para uma lista completa.
`description`	String	Não	Uma descrição legível sobre o que este ficheiro de simulação faz.
`common_args`	Array de Strings	Não	Uma lista ordenada de argumentos posicionais que vêm após o `simulation_type`. Para `application-uplink`, deve ser `["<application_id>", "<device_id>"]`. Para `gateway-status`, seria `["<gateway_id>", "<address>"]`.
`flags`	Objeto	Não	Um mapa chave-valor de flags da CLI a serem incluídas. A chave é o nome da flag *sem* o `--` à frente. Para flags booleanas que não recebem um valor (como `--confirmed`), use `true`. Para flags aninhadas, use a notação de ponto (ex: `"settings.data-rate-index"`).
`payload_source`	Objeto	Não	Define como o payload principal para a simulação é gerado. Se omitido, nenhum payload será gerado pelo script (útil para simulações como `gateway-status` que não têm um payload simples). Veja a secção 2.1 para mais detalhes.
`periodic_settings`	Objeto	Não	Configura o modo de envio periódico. Se omitido, a simulação só será executada manualmente com o comando `simulate`. Veja a secção 2.2 para mais detalhes.

2.1 O Objeto `payload_source`

Este objeto indica ao simulador como gerar os bytes do payload.

Chave	Tipo	Obrigatório?	Descrição

`type`	String	**Sim**	O método de geração de payload. Opções válidas são: - `json_template`: Gera um payload binário complexo usando um ficheiro de template separado (recomendado). - `random_int`: Gera um inteiro aleatório com um comprimento de bytes especificado. - `fixed_hex`: Usa uma string hexadecimal fixa.
`file`	String	Se `type` for `json_template`	O nome do ficheiro `payload_template.json` a ser utilizado.
`num_bytes`	Inteiro	Se `type` for `random_int`	O número de bytes para o inteiro aleatório (ex: `4` para um inteiro de 32 bits).
`value`	String	Se `type` for `fixed_hex`	A string hexadecimal a ser usada como payload (ex: `"CAFE01"`).

2.2 O Objeto `periodic_settings`

Este objeto controla a execução automática e repetida da simulação.

Chave	Tipo	Obrigatório?	Descrição
`interval`	Inteiro	**Sim**	O tempo, em segundos, entre cada uplink simulado.
`enabled_on_load`	Booleano	Não	Se definido como `true`, a simulação periódica iniciará automaticamente assim que este ficheiro de configuração for carregado com o comando `load_sim_config`. O padrão é `false`.

3. O Arquivo de Template de Payload (`payload_template.json`)

Este arquivo é o coração da geração de payloads complexos. Ele define a estrutura binária exata dos seus dados.

Exemplo:

```
{
  "_field_order": ["temperature", "humidity"],
  "fields": {
    "temperature": {
      "type": "int",
      "packer": "h",
      "byte_order": "big"
    },
    "humidity": {
```

```
      "type": "uint",
      "packer": "B"
    }
  }
}
```

Campos de Nível Superior

Chave	Tipo	Obrigatório?	Descrição
`_field_order`	Array de Strings	**Sim**	Define a ordem exata em que os campos serão empacotados no array de bytes final. A ordem nesta lista é crítica para criar um payload binário válido.
`fields`	Objeto	**Sim**	Um mapa chave-valor onde cada chave é um nome de campo de `_field_order` e cada valor é um objeto que define as propriedades desse campo.

Propriedades de Definição de Campo

Estas propriedades são definidas para cada campo dentro do objeto `fields`.

Propriedade	Tipo	Descrição
`type`	String	**(Obrigatório)** O tipo de dados a ser gerado. **Valores:** `int`, `uint`, `float`, `string`, `hex_string`, `choice`.
`packer`	String	Um caractere de formato do módulo `struct` do Python que define como empacotar o valor em bytes. Esta é a propriedade mais importante para criar um formato binário correto. Consulte a Documentação do `struct` do Python para todas as opções. **Valores comuns:** `B` (uint de 1 byte), `h` (int de 2 bytes), `H` (uint de 2 bytes), `i` (int de 4 bytes), `I` (uint de 4 bytes), `f` (float de 4 bytes), `d` (double de 8 bytes). Para strings de comprimento fixo, use um formato como `4s` (string de 4 bytes).
`byte_order`	String	Para formatos `packer` de múltiplos bytes (`h`, `H`, `i`, `I`, `f`, etc.). **Valores:** `"big"` (ordem de rede, byte mais significativo primeiro) ou `"little"`. O padrão é `big`.
`min`, `max`	Número	Para os tipos `int`, `uint` e `float`. Define o intervalo inclusivo para a geração de valores aleatórios. Os padrões são baseados no tamanho do `packer` se não forem fornecidos.
`precision`	Inteiro	Para o tipo `float`. O número de casas decimais para arredondar o float gerado aleatoriamente.

`length`	Inteiro	Para o tipo `string` (quando nenhum `packer` é usado). O número de caracteres a serem gerados.
`length_bytes`	Inteiro	Para o tipo `hex_string`. O número de *bytes* que a string hexadecimal final deve representar (a própria string terá o dobro do comprimento).
`charset`	String	Para o tipo `string`. O conjunto de caracteres a ser usado para a geração aleatória. **Valores:** `alphanumeric`, `alnum`, `hex`, `ascii`.
`values`	Array ou Objeto	Para o tipo `choice`. Fornece as opções para escolher aleatoriamente. - **Array:** Uma lista de valores possíveis (ex: `[10, 20, 30]`). - **Objeto:** Um mapa de nomes legíveis para o valor numérico que deve ser empacotado (ex: `{"ON": 1, "OFF": 0}`).
`encoding`	String	Para o tipo `string` (quando nenhum `packer` é usado). A codificação a ser usada ao converter a string gerada em bytes. O padrão é `"utf-8"`.
`_comment`	String	Um lugar para adicionar notas legíveis. Este campo é ignorado pelo script.