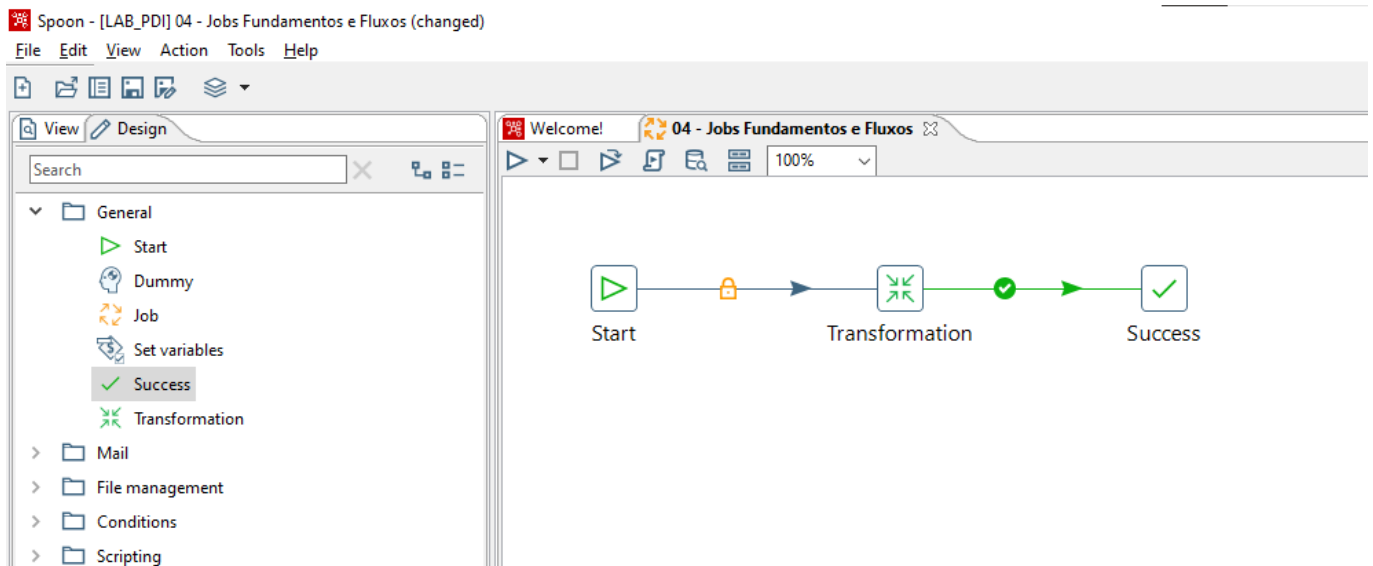


Exercício 04 – Jobs

- 4.1** Crie uma nova transformação, chamada “04 – Jobs Fundamentos”, gere 10.000 linhas de input e jogue o resultado em um dummy:



- 4.2** Agora será criado um elemento novo, o “Job”, que deve ser salvo como “04 – Jobs Fundamentos e Fluxos”. Na sessão “Design”, traga os steps “Start”, “Transform” e “Success”:



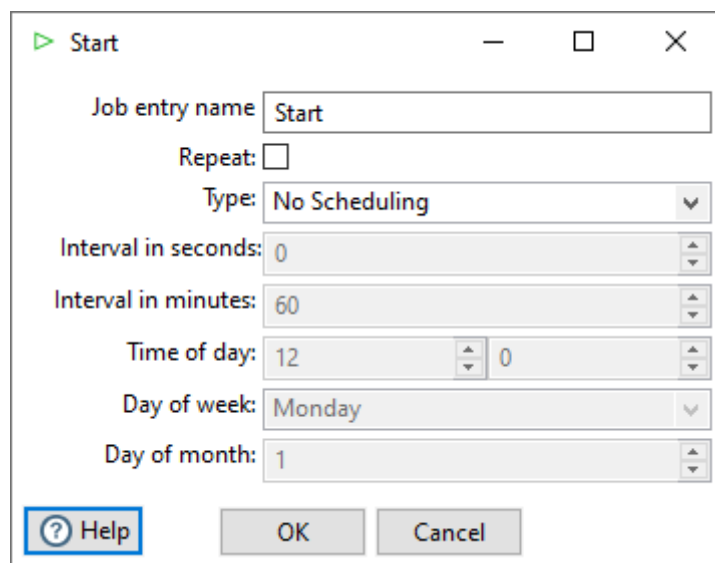
- 4.3** No step “Start”, nada precisa de ser configurado. É o step que funciona como um marcador para início dos fluxos de qualquer Job, e permite também o agendamento utilizando o agendador do próprio Pentaho. Para nosso exercício, não será necessário realizar configurações para este step:



Detalhamento Step:

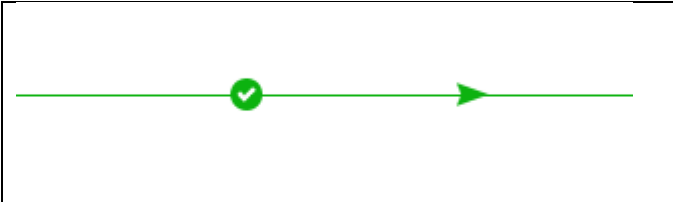
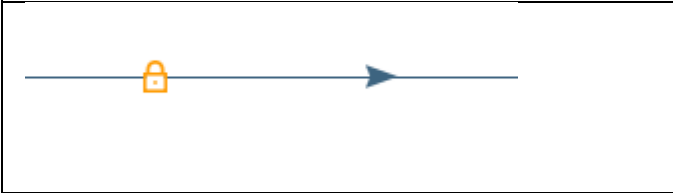
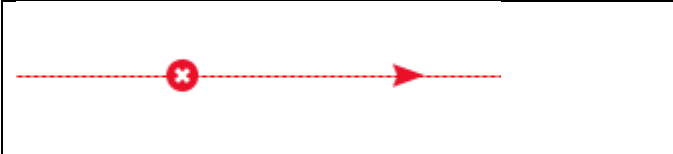
Repeat – quando marcado faz com que o job inicie automaticamente uma nova execução sempre que uma é concluída com sucesso, e não já uma data para encerramento.

Type – escolha sobre a forma com qual o job será automatizado, em um intervalo de minutos ou segundos, diariamente, semanalmente ou mensalmente.



Exercício 05 – Jobs HOPS Fluxos

5.1 De forma similar aos HOPs de transformação, o Pentaho integra um step de job ao outro por meio de HOPs, com algumas possibilidades a mais

	<p>HOP Verde: fluxo vai passar para o próximo step se e somente se o anterior der ok.</p>
	<p>HOP Cadeado Amarelo: indica que o fluxo irá passar para o próximo step independente do estado do hop anterior.</p>
	<p>O HOP vermelho permite que um step seja acionamento se e somente se o step anterior acusar erro.</p>

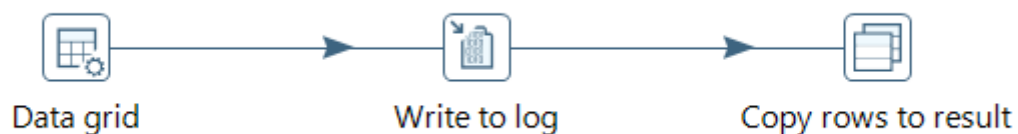
Exercício 06 – Jobs Loops e Variáveis

6.1 Este exercício consiste na utilização de Jobs e Transformações para realização de processos em loop, de forma similar ao funcionamento de cursores SQL, nos quais uma lista de parâmetros é repassada para o código do cursor, que executa toda sua lógica para cada um dos itens de entrada. Será composto por 2 Jobs e 2 Transformações, cada um com uma função específica. A seguir, uma visão geral destes 4 elementos:

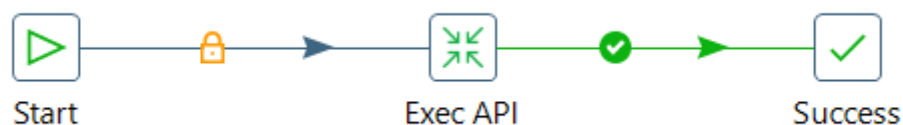
6.1.1 Job principal, “06 – Jobs API Loop”, a partir do qual o processo completo será instanciado:



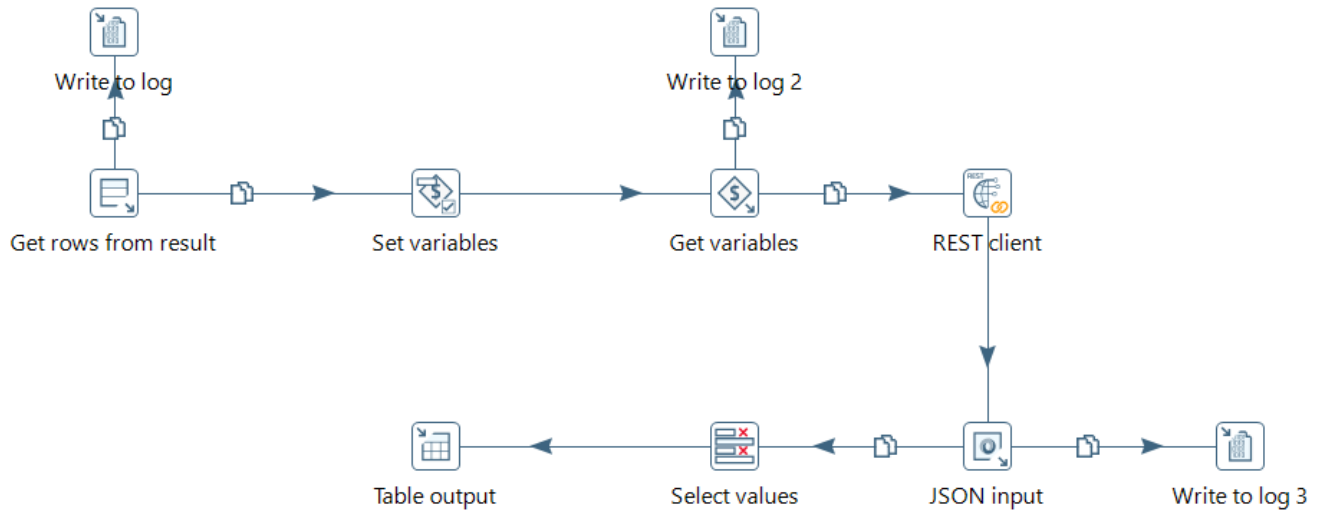
6.1.2 A primeira transformação, “Seta UF”, irá disparar a transformação “06 – Jobs Seta UF”:



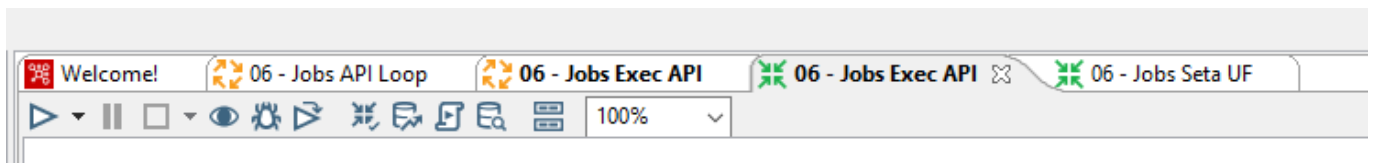
6.1.3 O segundo Job, “Executa API”, irá disparar o Job “06 – Jobs Exec API”:



6.1.4 Dentro do Job, “06 – Jobs Exec API”, será disparada a transformação “06 – Jobs Exec API”:



6.1.5 Ao final, todos estes elementos deverão ser construídos e validados:

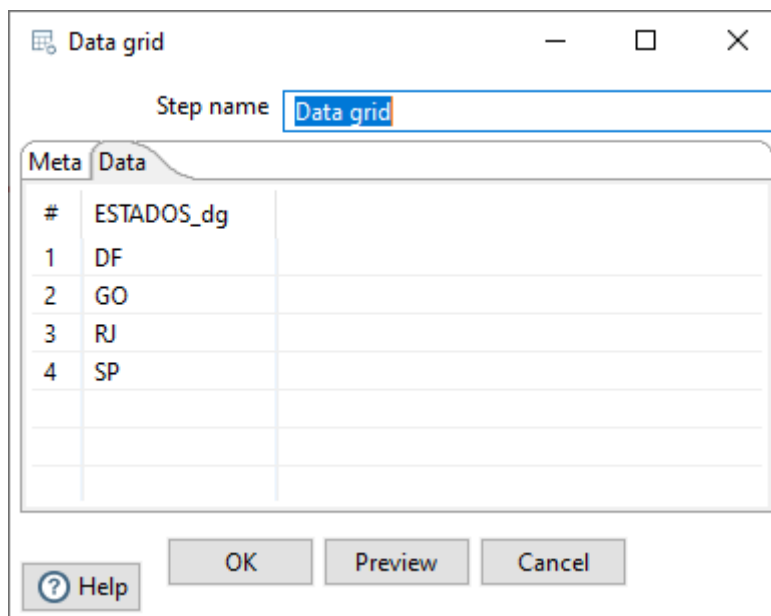


6.2 Crie um job chamado “06 – Jobs API Loop”, e adicione os steps “Start”, “Tranformation”, “Job” e “Success”.



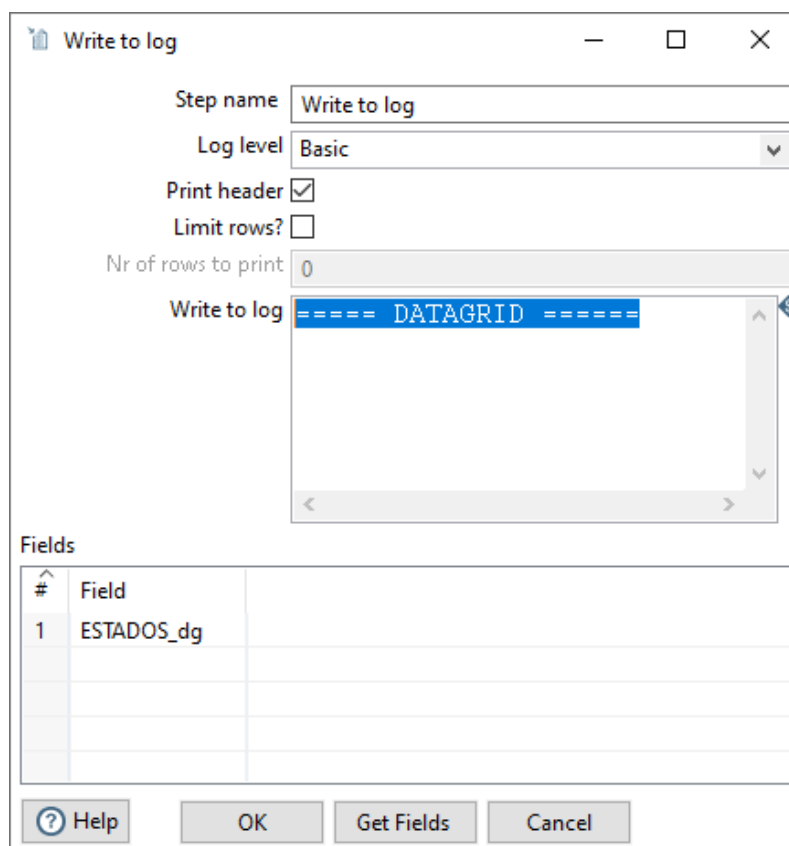
6.3 Os steps “Start” e “Success” não necessitam de configuração, a não ser que um agendamento seja feito, conforme descrito no item 4.3 deste documento. Salve o Job e passe para o próximo item do exercício.

6.4 Crie uma transformação chamada “06 – Jobs Seta UF”, e adicione os steps “Start”, “Tranformation”, “Job” e “Success”. Crie uma lista estática com os seguintes valores:



#	ESTADOS_dg
1	DF
2	GO
3	RJ
4	SP

6.5 Adicione o step “Write to Log”, que irá permitir a visualização do log desta transformação, a partir do monitor de execução do Job principal “06 – Jobs API Loop”. Adicione o campo cujo valor se deseja imprimir em tela, e também um comentário na caixa “Write to log”, para facilitar a localização.



Step name: Write to log

Log level: Basic

Print header: ☒

Limit rows?: ☐

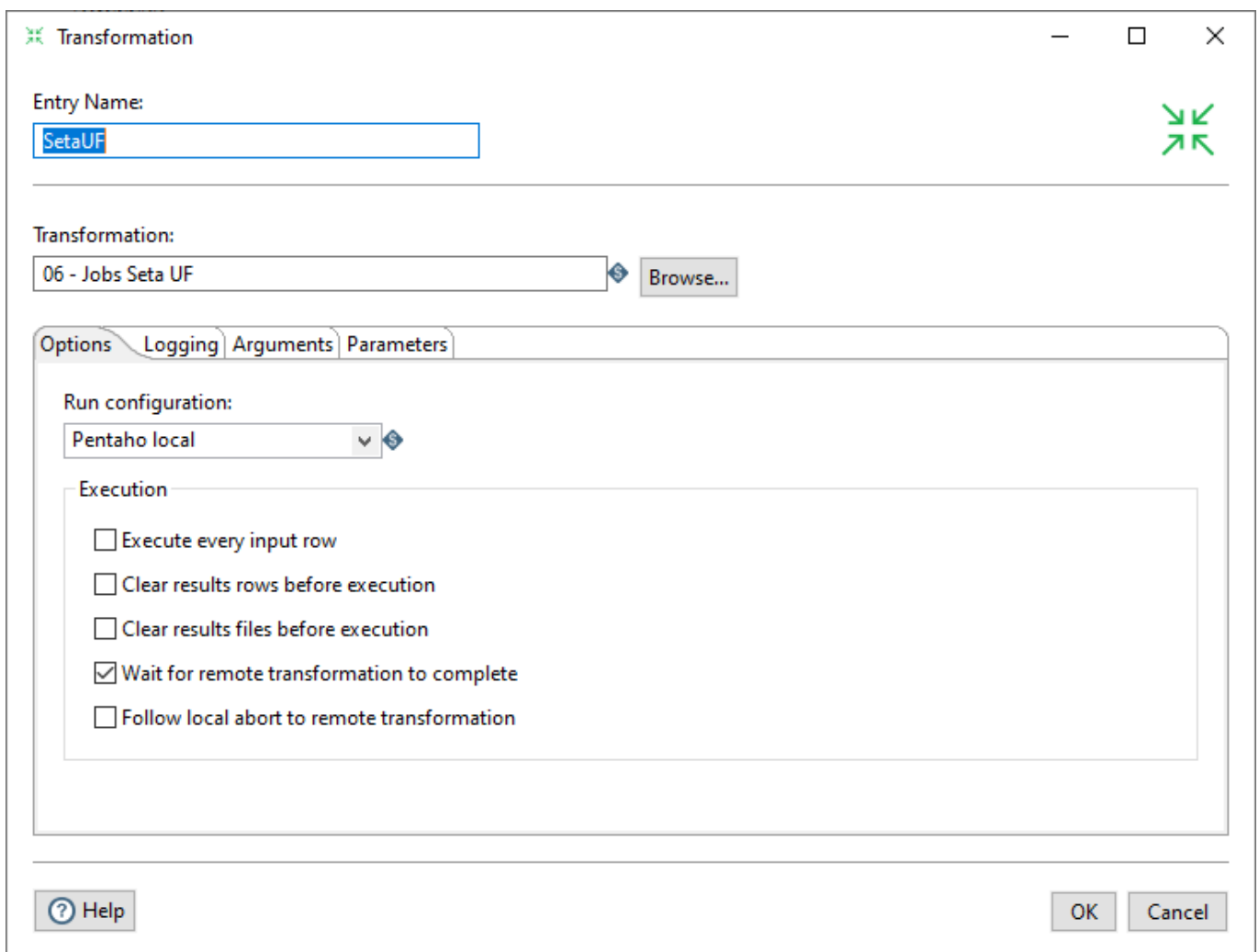
Nr of rows to print: 0

Write to log: ===== DATAGRID =====

#	Field
1	ESTADOS_dg

6.6 Adicione o step “Copy rows to result”, para levar o conteúdo da transformação para um outro elemento fora desta transformação. No caso, a transformação irá encerrar a partir do step “Copy rows to result” (que não precisa de configurações), e irá deixar em memória o seu conteúdo, que será configurado para recebimento em outra transformação. Salve esta transformação e passe para o próximo item deste exercício.

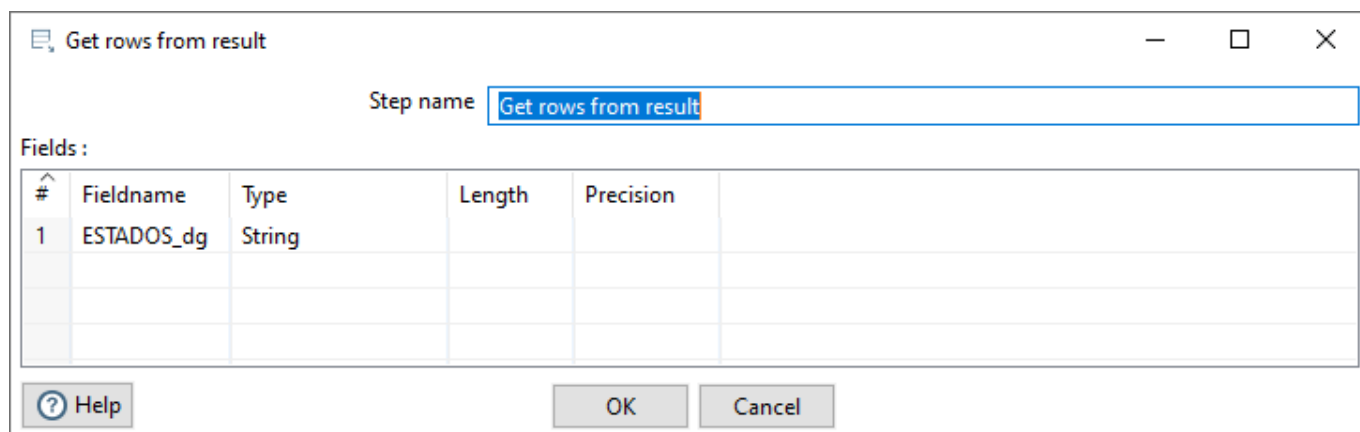
6.7 Volte ao Job principal, e edite o step “Seta UF”, para apontá-lo para a transformação “06 – Jobs Seta UF”, recém criada. Salve a transformação e vá para o próximo passo do exercício:



The screenshot shows the "Transformation" configuration window in Pentaho. The "Entry Name" field is set to "SetaUF". The "Transformation" dropdown is set to "06 - Jobs Seta UF", with a "Browse..." button next to it. The "Options" tab is selected, showing the "Run configuration:" dropdown set to "Pentaho local". Under the "Execution" section, the checkbox "Wait for remote transformation to complete" is checked, while the others are unchecked. At the bottom, there are "Help", "OK", and "Cancel" buttons.

6.8 Crie um novo Job, e o salve como “06 – Jobs Exec API”, e adicione os campos obrigatórios “Start” e “Success”, além de um step de “Transformation”, e vá par ao próximo item do exercício:

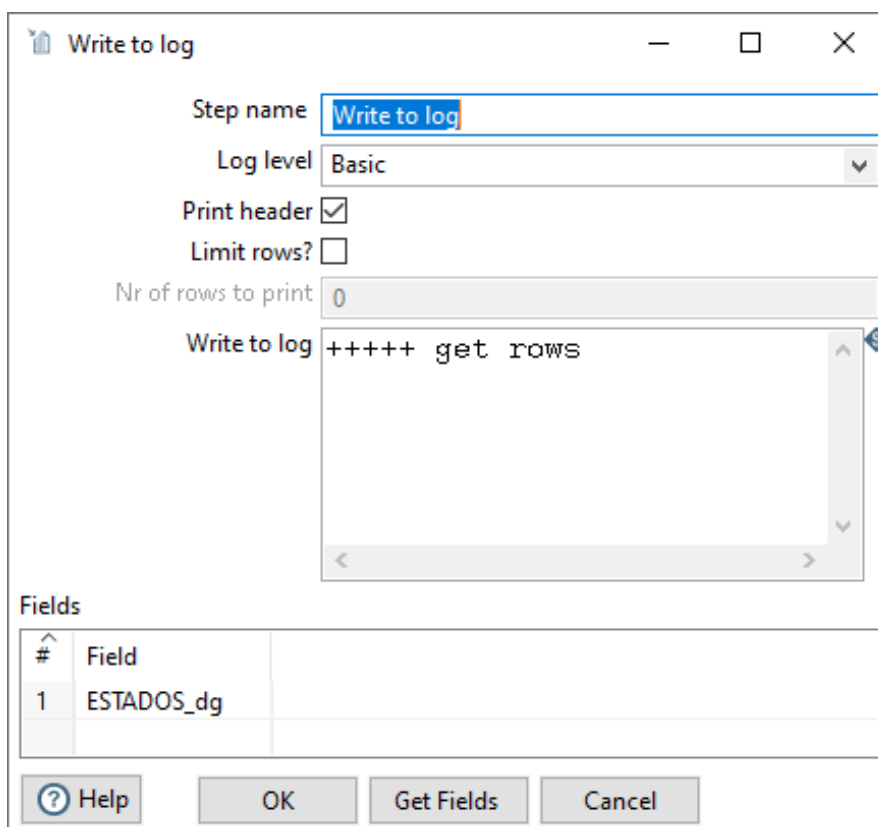
6.9 Crie uma nova transformação, chamada “06 – Jobs Exec API”, e adicione o step “Get rows from result”, para poder receber os dados do fluxo de execução de outra transformação, conforme definido no item 6.6:



#	Fieldname	Type	Length	Precision
1	ESTADOS_dg	String		

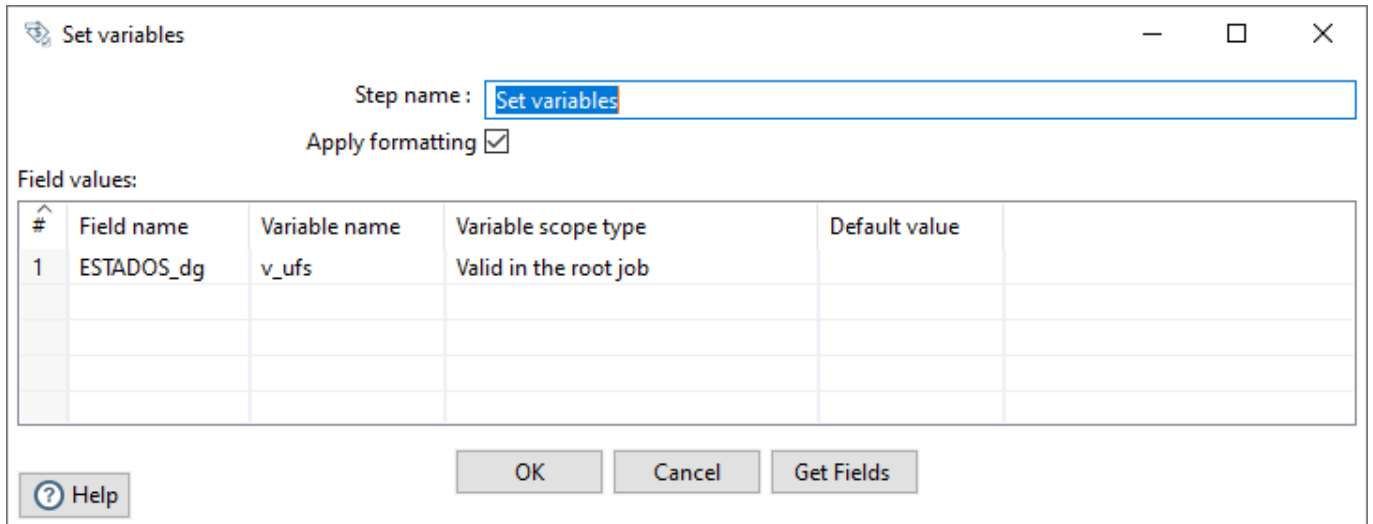
Obs.: O campo definido na seção “Fields”, coluna “Fieldname” deve ser inserido manualmente, e com o mesmo nome utilizado na configuração de saída do item 6.6 .

6.10 Adicione um step “Write to log”.



#	Field
1	ESTADOS_dg

6.11 Adicione um step “Set variables”, para declarar a variável que irá armazenar a lista de execução. Na seção “Field values”, campo “Field name”, insira o nome definido no step anterior. Em “Variable name”, defina o nome da variável que será repassada à API. Em “Variable scope type”, defina quais níveis de execução irão conseguir consumir a variável, sendo que em nosso caso, iremos marcar “Valid in the root job”:



#	Field name	Variable name	Variable scope type	Default value
1	ESTADOS_dg	v_ufs	Valid in the root job	

Obs.: Os escopos de variável que poderão ser escolhidos são:

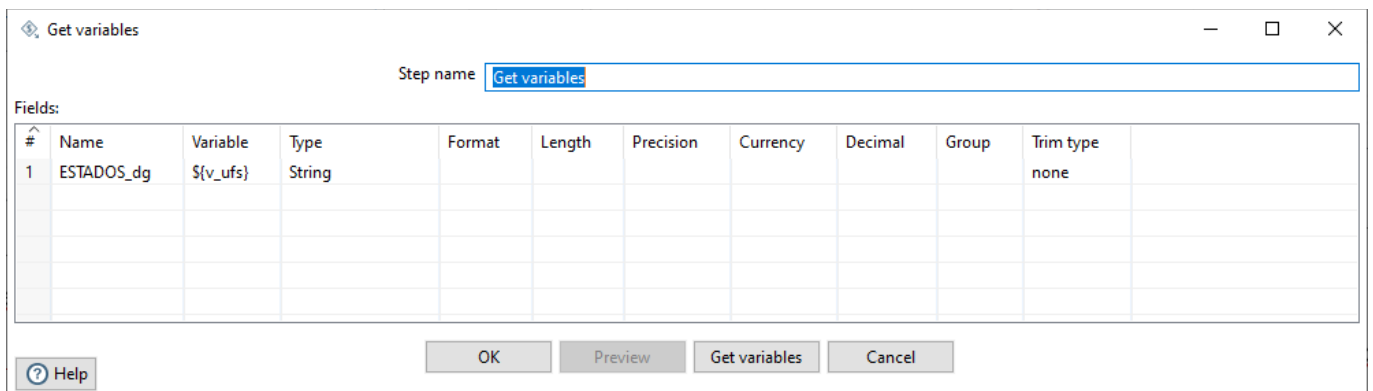
Valid in the root job – O job principal consome o conteúdo

Valid in the grand-parent job – O maior job da thread consome o conteúdo

Valid in the parent job – Apenas o job anterior consome o conteúdo

Valid in the Java Virtual Machine – Conteúdo na memória do Java, para acesso fora o pentaho

6.12 Adicione um step “Get variables”, para preencher o conteúdo da variável. Dê um nome ao campo que irá acomodar a variável, e preencha em “Variable” o nome da variável definida no step anterior, dentro da estrutura de ponteiros `${<nome da variável>}`, sendo que no caso deste exercício, o valor a ser preenchido nesta coluna será `${v_ufs}`:

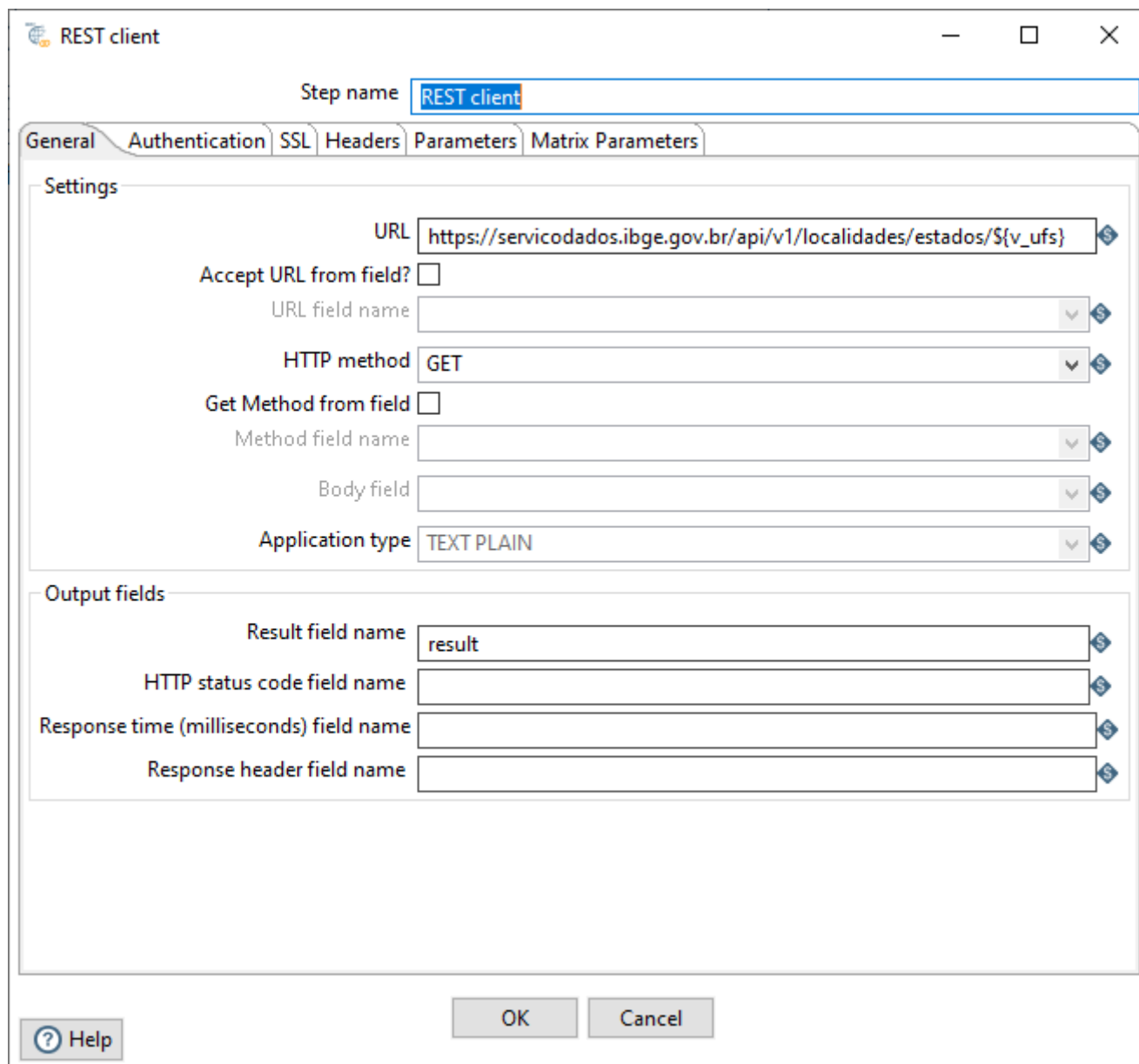


#	Name	Variable	Type	Format	Length	Precision	Currency	Decimal	Group	Trim type
1	ESTADOS_dg	\${v_ufs}	String							none

Obs.: Adicione também um step de gravação de log.

6.13 Adicione um step “REST Client”, e o configure para receber dados da API do IBGE (<https://servicodados.ibge.gov.br/api/v1/localidades/estados/>), adicionando como parâmetro ao seu final, a variável criada anteriormente:

URL: [https://servicodados.ibge.gov.br/api/v1/localidades/estados/\\${v_ufs}](https://servicodados.ibge.gov.br/api/v1/localidades/estados/${v_ufs})



REST client

Step name: REST client

General | Authentication | SSL | Headers | Parameters | Matrix Parameters

Settings

URL: [https://servicodados.ibge.gov.br/api/v1/localidades/estados/\\${v_ufs}](https://servicodados.ibge.gov.br/api/v1/localidades/estados/${v_ufs})

Accept URL from field? ☐

URL field name:

HTTP method: GET

Get Method from field? ☐

Method field name:

Body field:

Application type: TEXT PLAIN

Output fields

Result field name: result

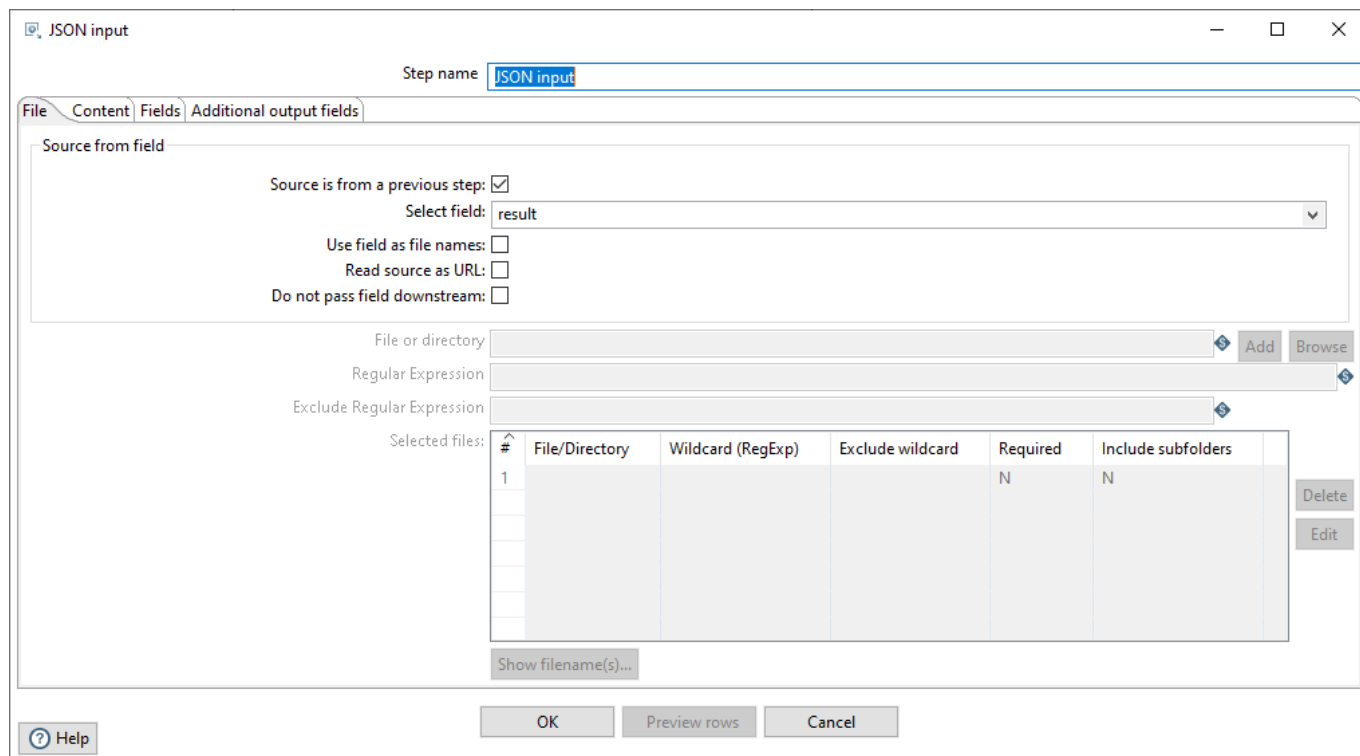
HTTP status code field name:

Response time (milliseconds) field name:

Response header field name:

Help OK Cancel

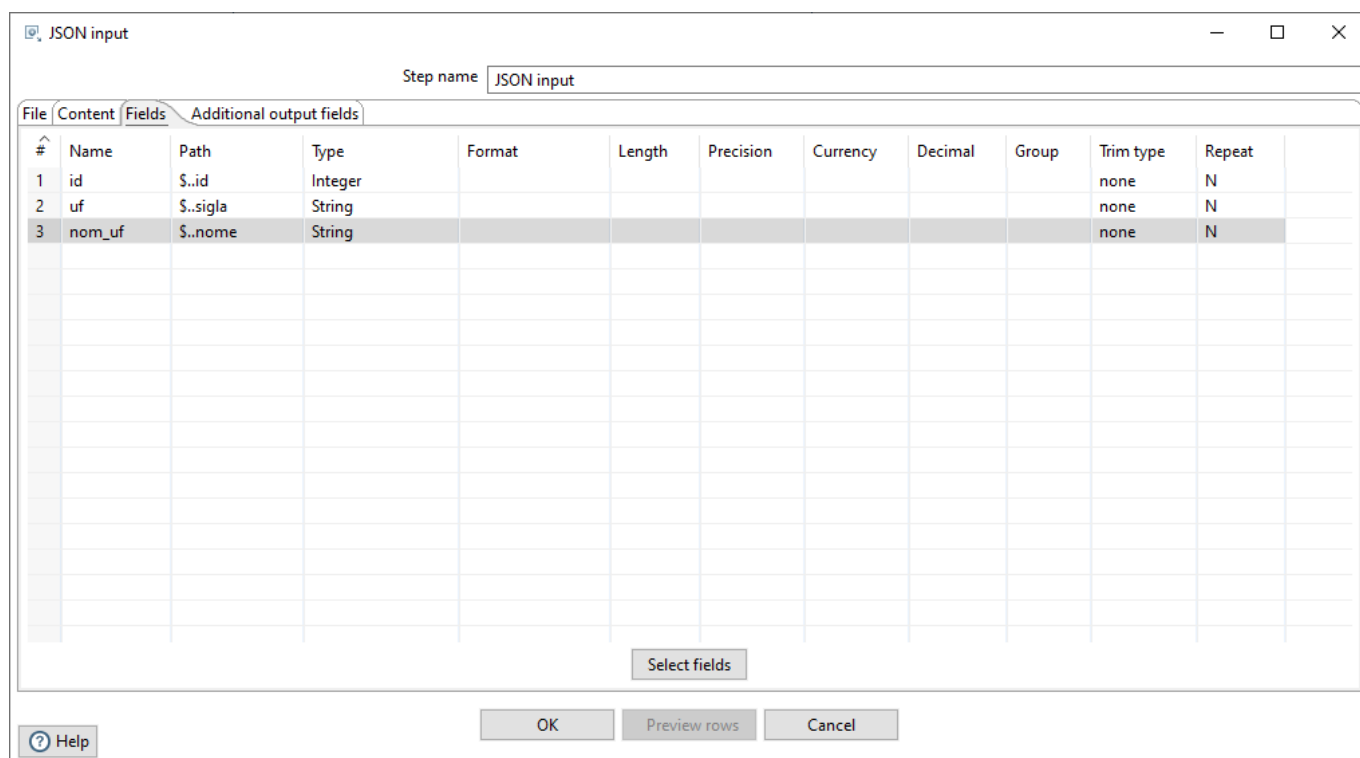
6.14 Adicione um step “JSON input”, e na aba “File”, marque a opção “Source is from a previous step” e indique o campo de origem dos dados. Já na aba “Fields”, configure manualmente os campos de entrada provenientes do resultado do REST:



The screenshot shows the 'JSON input' step configuration window with the 'File' tab selected. The 'Step name' is 'JSON input'. Under 'Source from field', 'Source is from a previous step' is checked, and 'Select field' is set to 'result'. Other options like 'Use field as file names', 'Read source as URL', and 'Do not pass field downstream' are unchecked. Below, there are fields for 'File or directory', 'Regular Expression', and 'Exclude Regular Expression'. A table titled 'Selected files' is shown with one row:

#	File/Directory	Wildcard (RegExp)	Exclude wildcard	Required	Include subfolders
1				N	N

Buttons for 'Delete' and 'Edit' are next to the table. At the bottom are 'OK', 'Preview rows', and 'Cancel' buttons.



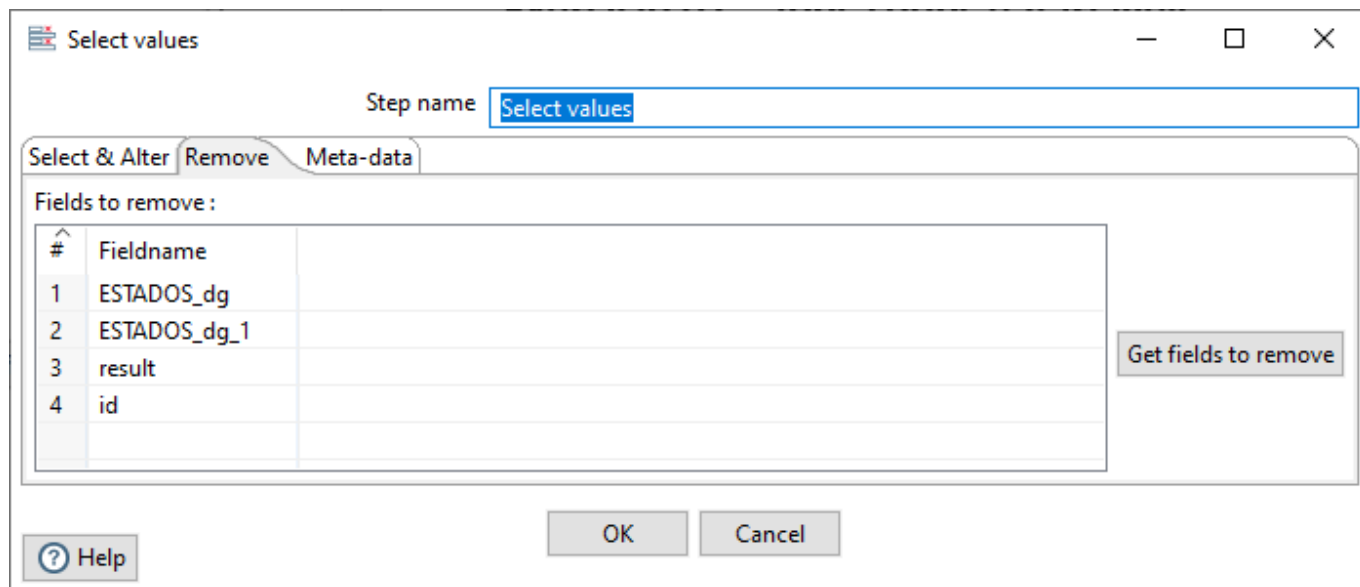
The screenshot shows the 'JSON input' step configuration window with the 'Fields' tab selected. The 'Step name' is 'JSON input'. A table lists the fields to be imported:

#	Name	Path	Type	Format	Length	Precision	Currency	Decimal	Group	Trim type	Repeat
1	id	\$.id	Integer							none	N
2	uf	\$.sigla	String							none	N
3	nom_uf	\$.nome	String							none	N

A 'Select fields' button is at the bottom. At the bottom of the window are 'OK', 'Preview rows', and 'Cancel' buttons.

Obs.: Adicione também mais um step de gravação em log.

6.15 Selecione apenas o campos necessários na saída do JSON:



Step name:

Select & Alter Remove Meta-data

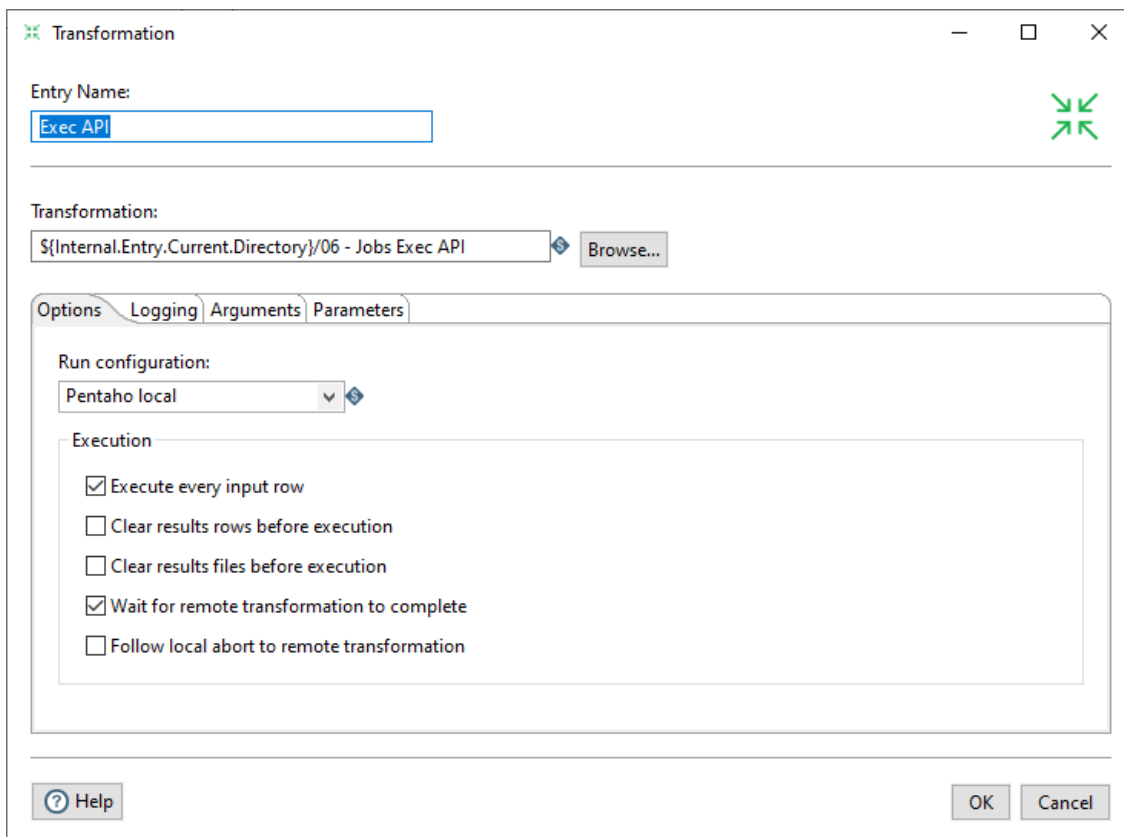
Fields to remove :

#	Fieldname
1	ESTADOS_dg
2	ESTADOS_dg_1
3	result
4	id

Get fields to remove

Help OK Cancel

6.16 Grave o resultado em uma tabela chamada “api_loop”, salve a transformação, e volte para o job “06 – Jobs Exec API”, configurando o step de job para transformação “Exec API” , apontando para invocar a transformação “06 – Jobs Exec API” recém criada. Na configuração, marque “Execute every input row”, para permitir o loop. Salve o Job.



Transformation

Entry Name:

Transformation: Browse...

Options Logging Arguments Parameters

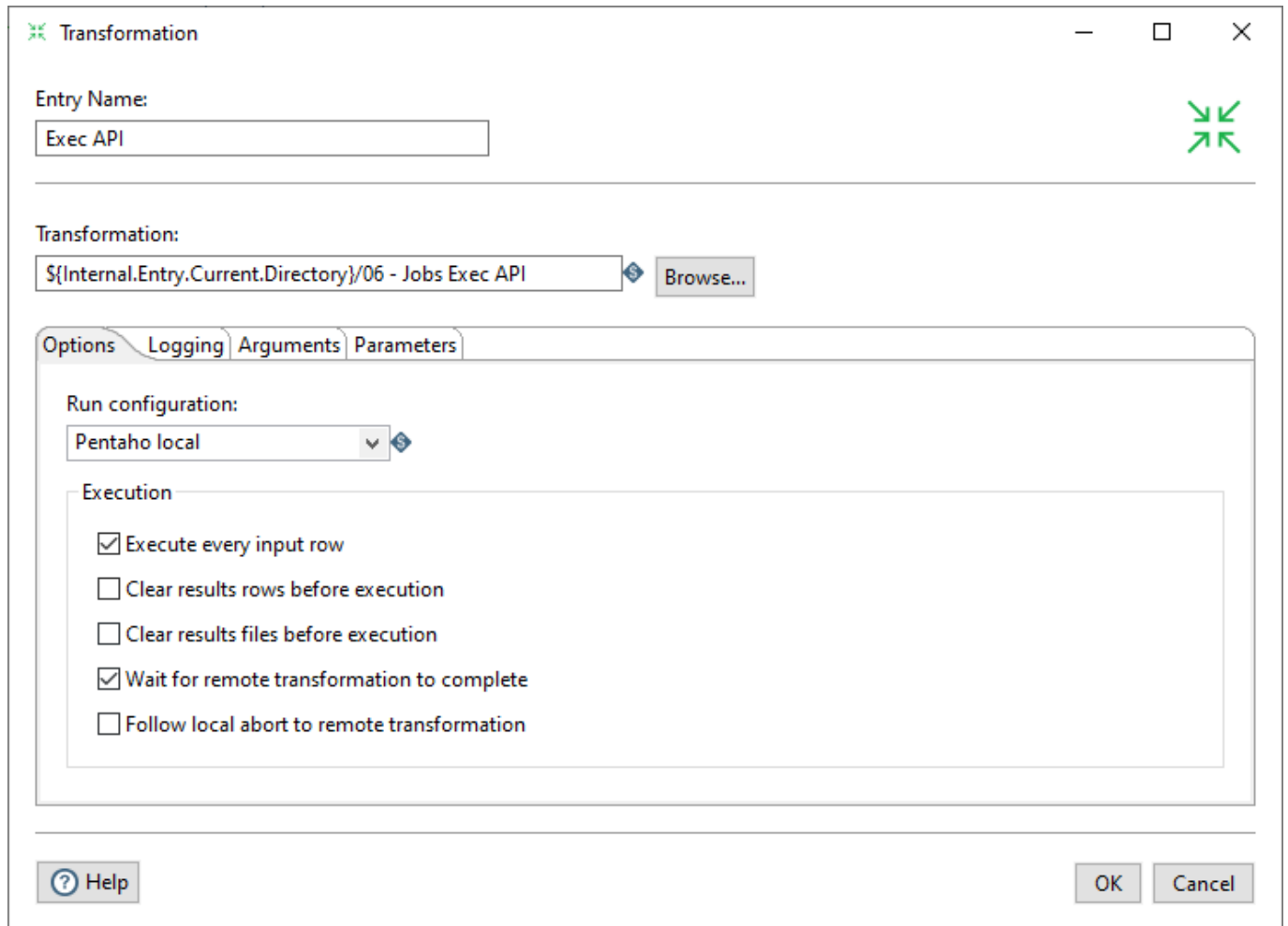
Run configuration:

Execution

- ☒ Execute every input row
- ☐ Clear results rows before execution
- ☐ Clear results files before execution
- ☒ Wait for remote transformation to complete
- ☐ Follow local abort to remote transformation

Help OK Cancel

6.17 Grave o resultado em uma tabela chamada “api_loop”, salve a transformação, e volte para o job “06 – Jobs Exec API”, configurando o step de job para transformação “Exec API”, apontando para invocar a transformação “06 – Jobs Exec API” recém criada. Na configuração, marque “Execute every input row”, para permitir o loop. Salve o Job.

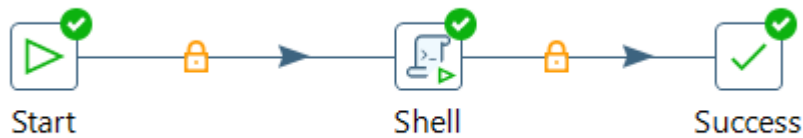


The screenshot shows the 'Transformation' configuration window in Pentaho. The 'Entry Name' is 'Exec API'. The 'Transformation' field is set to '\${Internal.Entry.Current.Directory}/06 - Jobs Exec API'. The 'Options' tab is selected, showing the 'Run configuration' dropdown set to 'Pentaho local'. Under the 'Execution' section, the following options are checked: 'Execute every input row', 'Wait for remote transformation to complete'. The other options, 'Clear results rows before execution', 'Clear results files before execution', and 'Follow local abort to remote transformation', are unchecked. The window has 'Help', 'OK', and 'Cancel' buttons at the bottom.

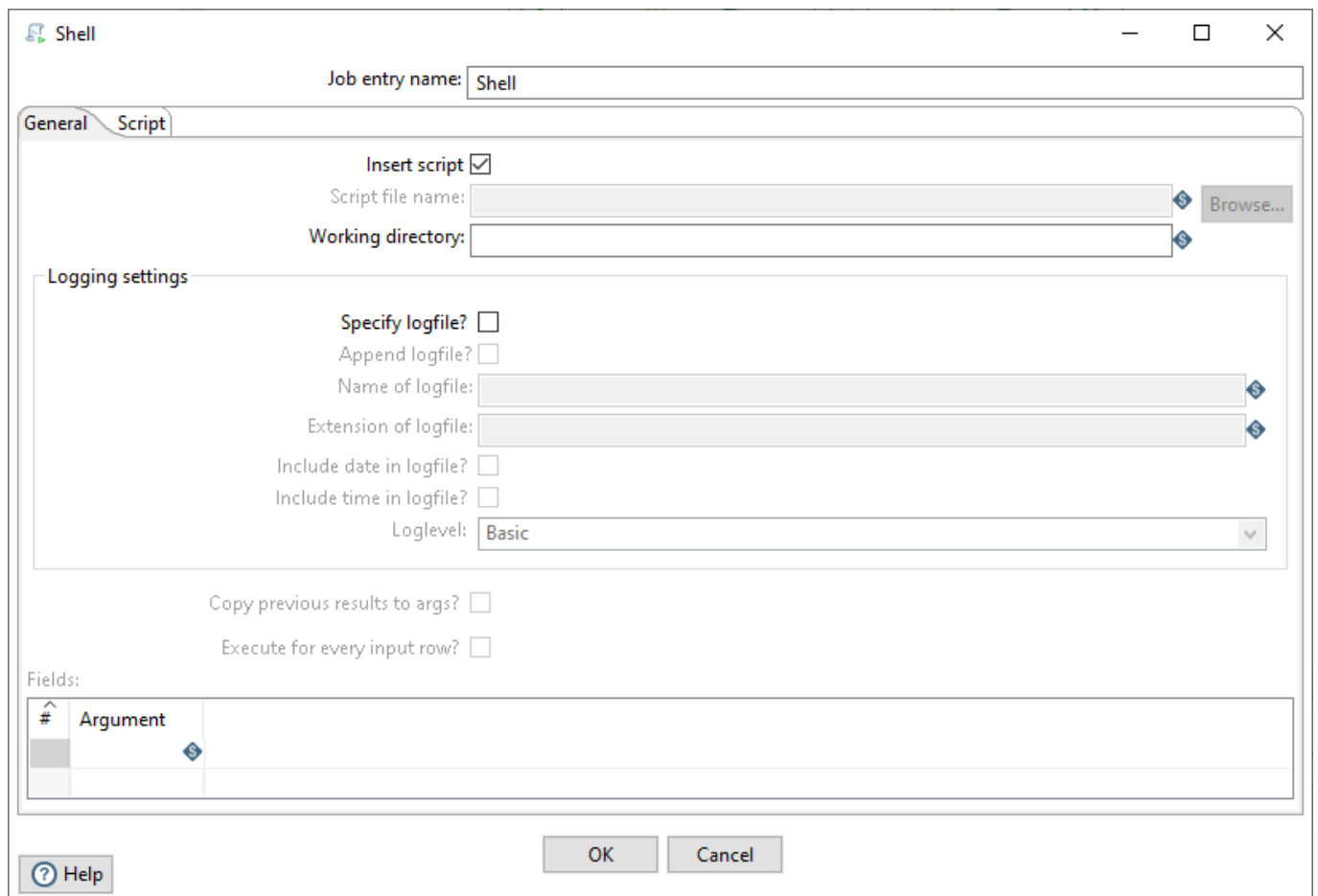
6.18 Retorne ao job principal “06 – Jobs API Loop” e o execute. Observe no log a passagem dos parâmetros, e verifique a carga da tabela.

Exercício 07 – Jobs Shell

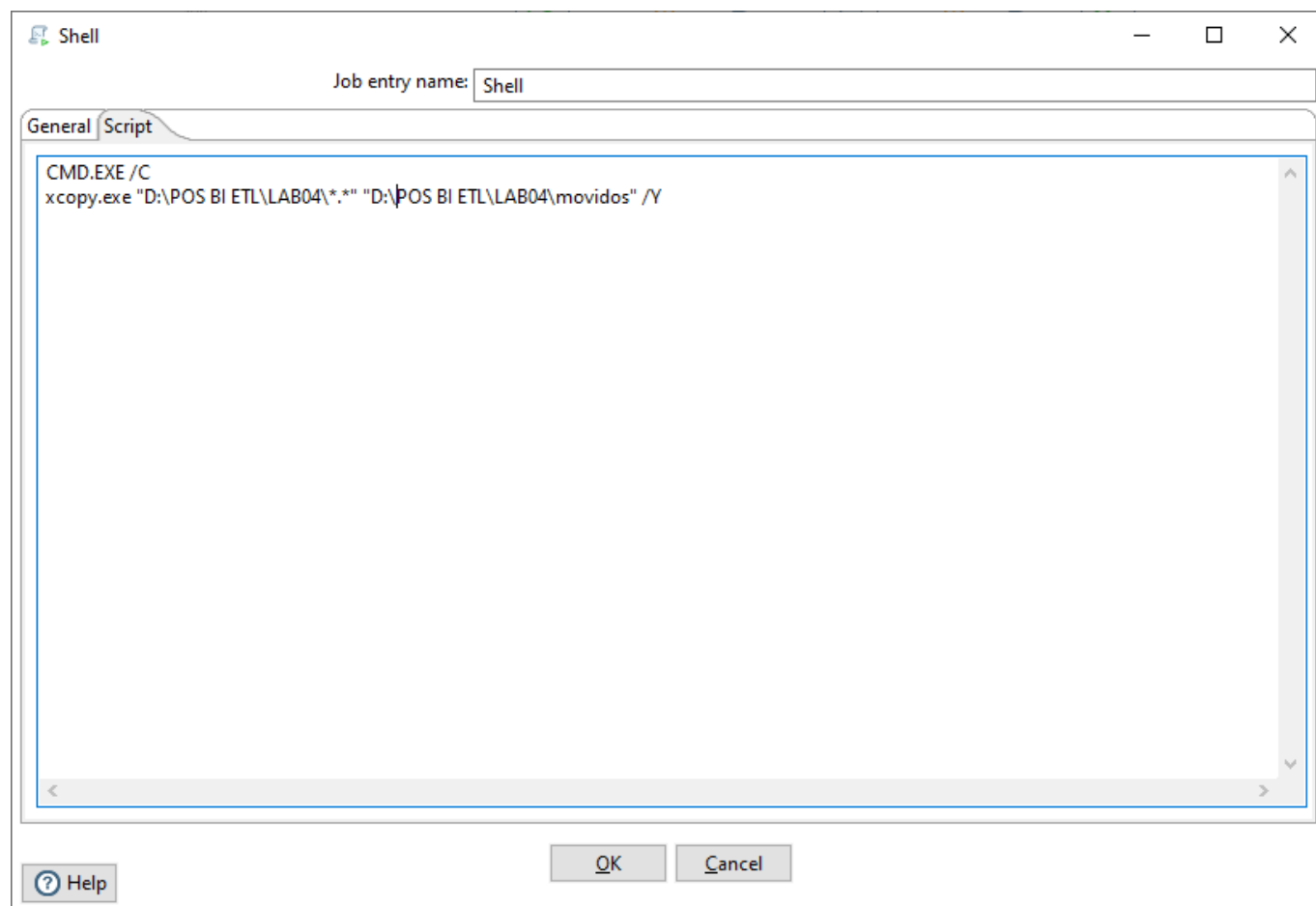
7.1 Cria uma nova transformação, “07 – Jobs Shell” , adicione os steps necessários ao Job, e também adicione o step “Shell”, de acordo com a figura a seguir:



7.2 Na configuração do step “Shell”, marque a opção “Insert script”, para permitir a escrita de um novo código na aba “Script”:



Configuração geral do script



Inserção do script