O ECL Scheduler

Equipe de documentação de Boca Raton

ECL Scheduler

Equipe de documentação de Boca Raton

Sua opinião e comentários sobre este documento são muito bem-vindos e podem ser enviados por e-mail para <docfeedback@hpccsystems.com>

Inclua a frase **Feedback sobre documentação** na linha de assunto e indique o nome do documento, o número das páginas e número da versão atual no corpo da mensagem.

2023 Version 8.10.24-1

O ECL Scheduler

Introdução

O ECL Scheduler é um componente de processamento instalado na plataforma do HPCC System. Ele normalmente é inicializado junto com a plataforma.

Uma interface para o agendador está disponível através do ECL Watch. A interface do ECL Scheduler permite que você veja uma lista das workunits agendadas. O componente também pode acionar um evento. Um “Evento” é uma cconstante no formato string que nomeia o evento para interceptação.

Uma ferramenta de linha de comando, *scheduleadmin* , está disponível no servidor instalada em /opt/HPCCSystems/bin.

Agendamento ECL

O agendamento ECL oferece uma maneira para automatizar processos no código ECL ou para encadear os processos para que eles funcionem em sequência. Por exemplo, é possível criar o código ECL que supervisiona uma zona de entrada de arquivos para a chegada de um arquivo e, quando ele chega, realiza a distribuição aos nós do Thor, processa, compila um índice e depois o adiciona a um superarquivo.

Como Funciona

O ECL Scheduler é baseado em eventos. O ECL Scheduler (Agendador ECL) monitora uma lista de agendamentos que contém workunits e eventos registrados e executa quaisquer workunits associadas a um evento quando o evento é acionado.

Seu código ECL pode ser executado quando um evento for acionado, ou pode acionar um evento. Se você enviar o código contendo uma cláusula **WHEN** , o evento e a workunit são registrados na Lista de agendamento. Quando um evento é acionado, a workunit é compilada e executada. Quando a workunit é concluída, o ECL Scheduler (Agendador ECL) a remove da Lista de agendamento.

Por exemplo, se você enviar uma workunit usando **WHEN(‘Event1’,’MyEvent’, COUNT(2))** no local adequado, ela será executada duas vezes (o valor de **COUNT**) antes que o ECL Scheduler (Agendador ECL) a remova da lista de agendamento e que a workunit seja marcada como concluída.

Componente do ECL

Instalação e Configuração

O Agendador ECL é instalado juntamente com a plataforma do HPCC. Ele é iniciado e interrompido usando hpcc-init, da mesma forma que todos os outros componentes do HPCC.

Utilizando o ECL Scheduler

Declarações utilizadas na Linguagem ECL

As seguintes declarações da Linguagem ECL são utilizadas:

WHEN

O serviço **WHEN** executa a ação sempre que o evento for acionado. A opção **COUNT** , não obrigatória, especifica o número de eventos para acionar as instâncias da ação.

NOTIFY

A ação **NOTIFY** aciona o evento para que o serviço do fluxo de trabalho **WHEN** possa dar andamento nas operações que ele precisa executar.

EVENT

A função **EVENT** retorna um evento acionado, que pode ser usado no serviço do fluxo de trabalho **WHEN** ou na ação **NOTIFY** . EVENT não é uma declaração, mas um parâmetro para que o WHEN/NOTIFY descreva para que tipo de evento ele é usado.

CRON

A função **CRON** determina um evento de controladores de tempo a serem usados no serviço do fluxo de trabalho **WHEN** . Essa função é um sinônimo de **EVENT('CRON', time)**. O termo CRON individual não é uma declaração, mas um parâmetro para que o WHEN/NOTIFY descreva para que tipo de evento ele é usado.

WAIT

A função **WAIT** é uma constante de string com o EventName a ser aguardado. É muito usada como o serviço de fluxo de trabalho **WHEN** , mas pode ser usada no código de condição.

Monitorando Funções na Biblioteca Padrão (Standard Library-STD.File)

MonitorFile

A função **MonitorFile** cria um job do monitor de arquivos no DFU Server para um arquivo físico.

MonitorLogicalFileName

A função **MonitorLogicalFileName** cria um job do monitor de arquivos no DFU Server para um arquivo lógico.

DFUPlus: Opções do Monitor

```
    dfuplus action=monitor event=MyEvent
```

Observação : O monitoramento de arquivos no DFUServer (usando a Biblioteca padrão ou o DFUPlus) cria uma DFU Workunit . Durante o monitoramento, o estado da Workunit é *monitorado* e ao acionar o

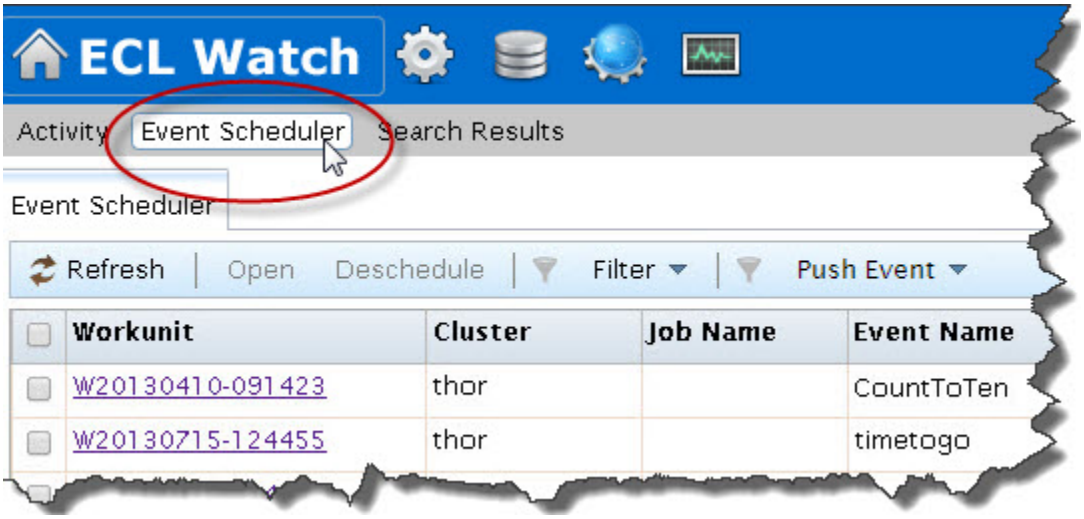
evento, ele é definido para *finalizado*. Você pode abortar uma WorkUnit DFU de monitoramento para parar o monitoramento do ECL Watch.

Interface no ECL Watch

Para acessar a interface do ECL Scheduler no ECL Watch, clique no **Event Scheduler** link localizado no submenu de navegação. A interface do Agendador será exibida juntamente com as workunit agendadas, se houver.

A lista de workunits agendadas possui duas colunas importantes, a **EventName** e a **EventText**.

Figure 1. Interface do ECL Scheduler



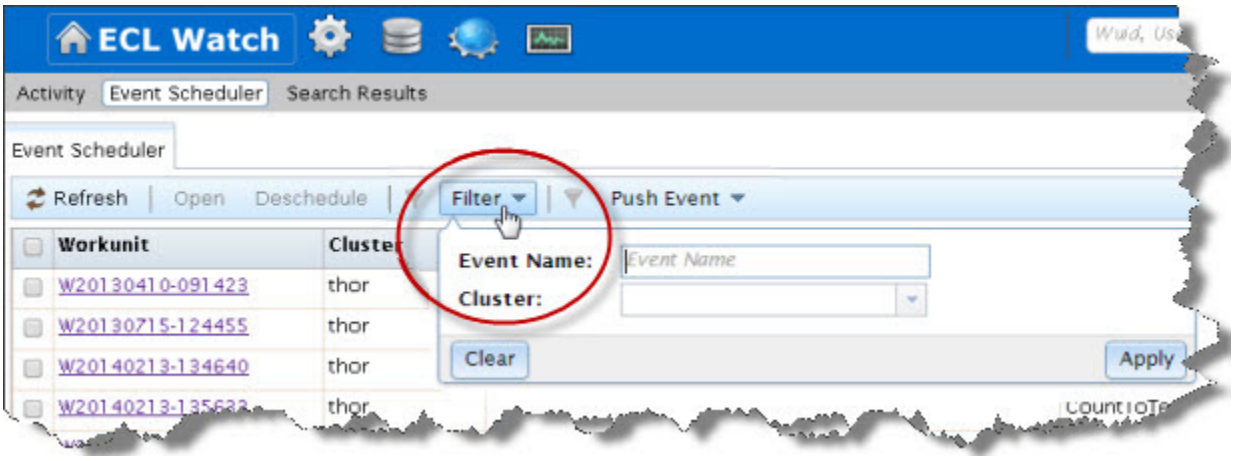
O EventName é criado no momento em que uma workunit é agendada. O EventText é um subevento que o acompanha.

Um evento pode ser acionado ao inserir o EventName e o Event Text nas caixas apropriadas, e clicando em seguida no botão **Push Event** . Trata-se do mesmo procedimento adotado ao acionar um evento usando NOTIFY.

Lista de Workunit do Agendador

A busca por workunit agendadas pode ser realizada por cluster ou EventName. Para filtrar por cluster ou por Event-Name, clique no botão de ação **Filter** . O submenu Filter será então exibido. Preencha os valores dos critérios de filtro (EventName ou cluster) e pressione o botão **Apply** . Ao especificar quaisquer opções de filtro, o botão de ação Filter exibe *Filter Set*.

Figure 2. Workunit na interface do Agendador



É possível classificar as workunits clicando no título da coluna.

Para ver os detalhes da tarefa, clique no link ID da workunit (WUID).

Você pode modificar as workunit agendadas na página Workunit Detail no ECL Watch. Selecione a página Workunit Details e pressione o botão **Reschedule** para agendar novamente uma workunit desagendada. Pressione o botão **Deschedule** para impedir que a workunit agendada seja executada. Também é possível acessar as opções Reshedule e Deschedule no menu contextual ao clicar em uma workunit com o botão direito

Se estiver usando a cláusula WHEN e contiver um número COUNT, quando for reagendada, a workunit reiniciará o COUNT de onde parou e continuará com o COUNT restante. Após o COUNT ter sido concluído pela tarefa, a opção de reagendamento não estará mais disponível.

Enviando Eventos

O Gerenciador de evento permite acionar ou “enviar” um evento para ajudar a gerenciar e testar suas workunit agendadas.

1. Pressione o botão de ação **Push Event** .

   A caixa de diálogo Push Event será aberta.

2. Digite o EventName (Nome do evento):

   O EventName é uma constante de strings, sem distinção entre maiúsculas e minúsculas, que nomeia o evento para interceptação.

   Ver também: EVENT

3. Insira o EventText:

   O EventText é uma constante de strings, sem distinção entre maiúsculas e minúsculas, que nomeia o tipo de evento específico para interceptação. Ele pode conter * e ? como elemento curinga.

   Ver também: EVENT

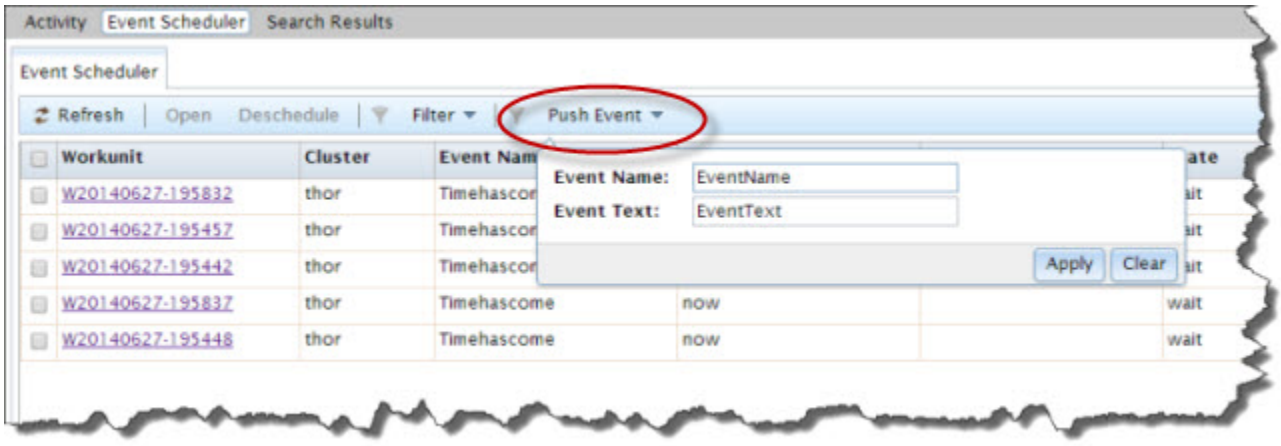
4. Pressione o botão **Apply** .

   Isso é o equivalente a

```
NOTIFY(EVENT(EventName,EventText));
```

   Ver também: NOTIFY, EVENT

Figure 3. PushEvent



Administrador do ECL Scheduler

Interface de Linha de Comando: scheduleadmin

O **scheduleadmin** é a interface da linha de comando do ECL Scheduler. Por padrão, o scheduleadmin está localizado no **/opt/HPCCSystems/bin/** em seu HPCC System.

scheduleadmin *daliserver operation* **[***options* **]**

daliserver	A URL (http:// ou https://) e/ou endereço IP do servidor Dali. A porta também deve ser incluída.
operation	Uma das seguintes ações: servers add remove removeall list monitor cleanup push
options	Opcional. Uma lista delimitada dos itens opcionais (listados abaixo) apropriados para executar a *operação* .

A aplicação **scheduleadmin** aceita parâmetros de linha de comando para manter a lista das tarefas monitoradas pelo ECL Scheduler.

Operações de Suporte

Há suporte para as seguintes operações.

Servidores

A operação do servidor retorna uma lista das filas do ECL Server anexadas ao servidor dali especificado que possuem eventos sendo monitorados.

Exemplo:

```
scheduleadmin 10.150.50.11:7070 servers

//returns data that looks like this:
eclserver_training
```

Adicionar wuid

A operação “adicionar” permite adicionar novamente a WUID especificada após ela ter sido removida da lista do monitor.

Essas opções são usadas pela operação add (adicionar):

WUID Um identificador de tarefa que contém uma ação do serviço de fluxo de trabalho WHEN .

Exemplo:

```
scheduleadmin 10.150.50.11 add W20120303-100635
```

Remover wuid

A operação “remover” permite remover a WUID especificada da lista do monitor.

Essas opções são usadas pela operação remove (remover):

WUID Um identificador de tarefa que contém uma ação do serviço de fluxo de trabalho WHEN .

Exemplo:

```
scheduleadmin 10.150.50.11 remove W20120303-100635
```

Removeall

A operação “remover todas” permite remover da lista do monitor todas as tarefas que contêm ações de serviços do fluxo de trabalho WHEN .

Exemplo:

```
scheduleadmin 10.150.50.11 removeall
```

List [*eclserver* | *event*]

A operação “listar” exibe a lista das workunits monitoradas e dos eventos que ainda irão ocorrer.

Essas opções são usadas pela operação List.

eclserver O nome de uma lista do ECL Server anexada ao servidor dali.

event Opcional. O nome de um evento. Se omitido, todos os eventos serão exibidos.

Exemplo:

```
scheduleadmin 10.150.50.11 list eclserver_training

 //returns data that looks like this:
 2012-03-16T19:18:40

 CRON
    10 19 * * *
       W20120316-130812

 MyEvent
    *
      W20120316-133145
```

Monitor[*eclserver*| *event*]

A operação “monitorar” bloqueia e atualiza a exibição da lista de tarefas monitoradas na medida em que as alterações ocorrem. Pressione a tecla ENTER para retornar ao prompt de comando.

As seguintes opções são usadas pela operação monitorar:

eclserver	O nome de uma lista do ECL Server anexada ao servidor dali.
event	Opcional. O nome de um evento. Se omitido, todos os eventos serão exibidos.

Exemplo:

```
scheduleadmin 10.150.50.11 monitor eclserver_training

 //returns data that looks like this:
 2012-03-16T19:07:22

 CRON
    40 18 * * *
       W20120316-124216
    10 19 * * *
    W20120316-130812
 monitoring...
```

Cleanup

A operação “limpeza” apara as ramificações não utilizadas da lista de árvores das unidades de trabalho monitoradas.

Exemplo:

```
scheduleadmin 10.150.50.11 cleanup
```

Push [*eclserver*| *event*]

A operação “push” publica o evento especificado como ocorrido. Isso possibilita "simular" a ocorrência de um evento para fins de teste.

As seguintes opções são usadas pela operação push:

event	O nome de um evento definido pelo usuário (NOT (NÃO) deve ser "CRON").
subtype	O valor da string a coincidir com o segundo parâmetro para a função EVENT.

Exemplo:

```
scheduleadmin 10.150.50.11 push MyFileEvent MyFile.d00
```

Uso do ECL

O ECL Scheduler (Agendador ECL) é uma ferramenta que pode desempenhar uma ação específica baseada em um evento específico. As seguintes funções podem ser visualizadas ou manipuladas no agendador.

WHEN

WHEN(*trigger*, *action* **[, BEFORE | SUCCESS | FAILURE])**

trigger	Um dataset ou ação que inicializa a *ação*.
action	A ação a ser executada.
BEFORE	Opcional. Especifica uma *ação* que deve ser executada antes da leitura da entrada.
SUCCESS	Opcional. Especifica uma *ação* especifica uma ação que só deve ser executada mediante ao SUCCESS do *trigger* (p.ex., quando os LIMITES não são excedidos).
FAILURE	Opcional. Especifica uma *ação* que só deve ser executada mediante a FALHA FAILURE do *acionador* (p.ex., quando o LIMITE LIMIT é excedido).

A função **WHEN** associa uma *ação* a um *acionador* (dataset ou ação) para que quando o *acionador* for executado, a *ação* também seja executada. Isso permite o agendamento de tarefas com base em acionadores.

Exemplo:

```
//a FUNCTION with side-effect Action
namesTable := FUNCTION
   namesRecord := RECORD
     STRING20 surname;
     STRING10 forename;
     INTEGER2 age := 25;
   END;
   o := OUTPUT('namesTable used by user <x>');
   ds := DATASET([{'x','y',22}],namesRecord);
   RETURN WHEN(ds,O);
END;

z := namesTable : PERSIST('z');
  //the PERSIST causes the side-effect action to execute only when the PERSIST is re-built
OUTPUT(z);
```

NOTIFY

[*attributename* := **] NOTIFY(** *event* **[**, *parm* **] [**, *expression* **])**

attributename	Opcional. O identificador desta ação.
event	A função EVENT , ou uma constante de string sem distinção entre maiúsculas e minúsculas que nomeia o evento a ser gerado.
parm	Opcional. Uma constante de string sem distinção entre maiúsculas e minúsculas que contém o parâmetro do evento.
expression	Opcional. Uma constante de string sem distinção entre maiúsculas e minúsculas que permite a especificação simples da mensagem para restringir o evento a uma workunit específica.

A ação **NOTIFY** aciona o *evento* para que a função WAIT function ou o serviço de fluxo de trabalho WHEN possa proceder com as operações que estão encarregador de operar.

O parâmetro *expression* permite definir um serviço em ECL iniciado por um *evento*, respondendo apenas para a tarefa que o iniciou.

Exemplo:

```
NOTIFY('testevent', 'foobar');

receivedFileEvent(STRING name) := EVENT('ReceivedFile', name);
NOTIFY(receivedFileEvent('myfile'));

//as a service
doMyService := FUNCTION
OUTPUT('Did a Service for: ' + 'EVENTNAME=' + EVENTNAME);
NOTIFY(EVENT('MyServiceComplete',
'<Event><returnTo>FRED</returnTo></Event>'),
EVENTEXTRA('returnTo'));
RETURN EVENTEXTRA('returnTo');
END;

doMyService : WHEN('MyService');
// and a call to the service
NOTIFY('MyService',
'<Event><returnTo>'+WORKUNIT+'</returnTo>....</Event>');
WAIT('MyServiceComplete');
OUTPUT('WORKUNIT DONE')
```

EVENT

EVENT(*event* , *subtype* **)**

event	Uma constante de strings, com distinção entre maiúsculas e minúsculas, que nomeia o evento para interceptação.
subtype	Uma constante de strings, com distinção entre maiúsculas e minúsculas, que nomeia o tipo de evento específico para interceptação. Pode conter * e ? como correspondência-curinga do subtipo do evento.
Return:	EVENT retorna um único evento.

A função **EVENT** retorna um evento acionado, que pode ser usado no serviço do fluxo de trabalho WHEN ou na ação NOTIFY . ou nas ações WAIT e NOTIFY.

Exemplo:

```
IMPORT STD;
MyEventName := 'MyFileEvent';
MyFileName  := 'test::myfile';

IF (STD.File.FileExists(MyFileName),
 STD.File.DeleteLogicalFile(MyFileName));
 //deletes the file if it already exists

STD.File.MonitorLogicalFileName(MyEventName,MyFileName);
 //sets up monitoring and the event name
 //to fire when the file is found

OUTPUT('File Created') : WHEN(EVENT(MyEventName,'*'),COUNT(1));
 //this OUTPUT occurs only after the event has fired

afile := DATASET([{ 'A', '0'}], {STRING10 key,STRING10 val});
OUTPUT(afile,,MyFileName);
 //this creates a file that the DFU file monitor will find
 //when it periodically polls


//*********************************
EXPORT events := MODULE
  EXPORT dailyAtMidnight := CRON('0 0 * * *');
  EXPORT dailyAt( INTEGER hour,
   INTEGER minute=0) :=
  EVENT('CRON',
   (STRING)minute + ' ' + (STRING)hour + ' * * *');
  EXPORT dailyAtMidday := dailyAt(12, 0);
END;
BUILD(teenagers): WHEN(events.dailyAtMidnight);
BUILD(oldies)  : WHEN(events.dailyAt(6));
```

CRON

CRON(*time* **)**

time	Uma expressão da string de caracteres contendo um tempo cron padrão unix.
Return:	CRON define um único evento de controladores de tempo.

A função **CRON** determina um evento temporal para uso no serviço WHEN ou função WAIT. Essa função é um sinônimo de EVENT('CRON', *time*).

O parâmetro de tempo segue o formato padrão de uma cron no unix, expressado em UTC (também conhecido como Tempo Médio de Greenwich) como uma string que contém os seguintes componentes de espaço delimitado:

minute hour dom month dow

minute	Um valor inteiro que significa o minuto da hora. Valores válidos de 0 a 59.
hour	Um valor inteiro que constitui a hora. Valores válidos de 0 a 23 (usando o relógio de 24 horas).
dom	Um valor inteiro que representa o dia do mês. Valores válidos de 1 a 31.
month	Um valor inteiro que representa o mês. Valores válidos de 1 a 12.
dow	Um valor inteiro que representa o dia da semana. Valores válidos de 0 a 6 (onde 0 representa o domingo).

Qualquer componente de *tempo* que você optar por não especificar, será substituído por um asterisco (*). Os intervalos de tempo devem ser definidos por um traço (-), as listas por uma vírgula (,), e “uma vez a cada n” usando a barra (/). Por exemplo, 6-18/3 no campo de hora acionará o controlador de tempo a cada três horas entre 6 da manhã e 6 da tarde, e 18-21/3,0-6/3 acionará o controlador de tempo a cada três horas entre 6 da tarde e 6 da manhã.

Exemplo:

```
EXPORT events := MODULE
  EXPORT dailyAtMidnight := CRON('0 0 * * *');
  EXPORT dailyAt( INTEGER hour,
   INTEGER minute=0) :=
  EVENT('CRON',
   (STRING)minute + ' ' + (STRING)hour + ' * * *');
  EXPORT dailyAtMidday := dailyAt(12, 0);
  EXPORT EveryThreeHours :=  CRON('0 0-23/3 * * *');
END;

BUILD(teenagers) : WHEN(events.dailyAtMidnight);
BUILD(oldies)    : WHEN(events.dailyAt(6));
BUILD(NewStuff)    : WHEN(events.EveryThreeHours);
```

WAIT

WAIT(*event***)**

event	Uma constante de string com o nome do evento a ser aguardado.

A ação **WAIT** é similar ao serviço de fluxo de trabalho WHEN, mas pode ser usada no código condicional.

Exemplo:

```
 //You can either do this:
action1;
action2 : WHEN('expectedEvent');

//can also be written as:
SEQUENTIAL(action1,WAIT('expectedEvent'),action2);
```

Monitoramento DFU e Relatórios

Os métodos a seguir são suportados pelo ECL Scheduler (Agendador ECL) e estão incluídos na ECL Standard Library Reference (Referência de biblioteca padrão ECL).

MonitorFile

STD.File.MonitorFile(*event,* **[** *ip* **] ,** *filename,* **[** *,subdirs* **] [** *,shotcount* **] [** *,espserverIPport* **])**

dfuwuid **:= STD.File.fMonitorFile(** *event,* **[** *ip* **] ,** *filename,* **[** *,subdirs* **] [** *,shotcount* **] [** *,espserverIPport* **]);**

event	Uma string terminada por nulo que contém o nome definido pelo usuário do evento a ser acionado quando *filename* aparecer. O valor é usado como o primeiro parâmetro da função EVENT.
ip	Opcional. Uma string terminada por nulo que contém o endereço IP do arquivo a ser monitorado. Normalmente, é uma zona de entrada de arquivos. Pode ser omitido apenas se o parâmetro *filename* contém uma URL completa.
filename	Uma string terminada por nulo que contém o caminho completo do arquivo a monitorar. Pode conter caracteres curinga (* e ?).
subdirs	Opcional. Um valor booleano que indica se devem ser incluídos os arquivos em subdiretórios que correspondem à máscara de curingas, quando *filename* contém curingas. Se omitido, o padrão é falso.
shotcount	Opcional. Um valor inteiro que indica o número de vezes que o evento deve ser gerado antes da conclusão do job de monitoramento. Um valor de um negativo (-1) indica que o job de monitoramento continuará até ser abortada manualmente. Se omitido, o padrão é 1.
espserverIPport	Opcional. Uma string terminada por nulo que contém o protocolo, o IP, a porta e o diretório ou DNS equivalente do programa do servidor' ESP. Normalmente são os mesmos IP e porta do ECL Watch, com "/FileSpray" anexado. Se omitida, o padrão é o valor contido no atributo lib_system.ws_fs_server.
dfuwuid	O nome do atributo que receberá a string terminada por nulo que contém o ID da workunit DFU (DFUWUID) gerado para o job de monitoramento.
Return:	fMonitorFile retorna uma string terminada por nulo que contém o ID da workunit DFU (DFUWUID).

A função **MonitorFile** cria uma workunit de monitoramento de arquivos no servidor do DFU. Após ser recebida, a tarefa entra no modo de monitoramento (que pode ser vista na tela da workunit DFU do ECL Watch), que sonda em intervalos fixos. Esse intervalo é especificado na configuração **monitorinterval** do servidor do DFU O intervalo padrão é 900 segundos (15 minutos). Se um arquivo com nome adequado chegar nesse intervalo, será acionado um *evento* com o nome do objeto acionador como subtipo do evento (consulte a função EVENT).

Esse processo continua até que:

1) O número do eventos *shotcount* tenham sido gerados

2) O usuário aborte a workunit DFU.

As funções STD.File.AbortDfuWorkunit e STD.File.WaitDfuWorkunit podem ser usadas para abortar ou aguardar o job DFU passando por elas e retornando ao *dfuwuid*.

Observe as seguintes advertências e restrições:

1) Os eventos são gerados apenas no início do job de monitoração ou posteriormente, de acordo com o intervalo de sondagem.

2) Observe que *evento* é gerado se o arquivo físico tiver sido criado desde o último intervalo de sondagem. Consequentemente, o *evento* poderá ocorrer antes de o arquivo ser fechado e os dados serem todos gravados. Para garantir que o arquivo não seja lido posteriormente antes de ser concluído, você deve usar uma técnica que elimine essa pos-

sibilidade. Por exemplo, use um arquivo separado de "indicador" em vez do próprio arquivo, ou renomeie o arquivo depois de criado e totalmente gravado.

3) Ao monitorar arquivos físicos, o parâmetro de subtipo da função EVENT (seu segundo parâmetro) corresponde ao URL completo do arquivo com um IP absoluto em vez do nome DNS/netbios do arquivo. Esse parâmetro não pode ser recuperado, mas pode ser usado apenas para corresponder um valor específico.

Exemplo:

```
EventName := 'MyFileEvent';
FileName  := 'c:\\test\\myfile';
LZ := '10.150.50.14';
STD.File.MonitorFile(EventName,LZ,FileName);
OUTPUT('File Found') : WHEN(EVENT(EventName,'*'),COUNT(1));
```

MonitorLogicalFileName

STD.File.MonitorLogicalFileName(*event,* *filename,* **[** , *shotcount* **] [** , *espserverIPport* **])**

dfuwuid **:= STD.File.fMonitorLogicalFileName(** *event,* *filename,* **[** , *shotcount* **] [** , *espserverIPport* **]);**

event	Uma string terminada por nulo que contém o nome definido pelo usuário do evento a ser acionado quando *filename* aparecer. O valor é usado como o primeiro parâmetro da função EVENT.
filename	Uma string terminada por nulo que contém o nome do arquivo lógico no DFU a ser monitorado.
shotcount	Opcional. Um valor inteiro que indica o número de vezes que o evento deve ser gerado antes da conclusão do job de monitoramento. Um valor de um negativo (-1) indica que o job de monitoramento continuará até ser abortada manualmente. Se omitido, o padrão é 1.
espserverIPport	Opcional. Uma string terminada por nulo que contém o protocolo, o IP, a porta e o diretório ou DNS equivalente do programa do servidor' ESP. Normalmente são os mesmos IP e porta do ECL Watch, com "/FileSpray" anexado. Se omitida, o padrão é o valor contido no atributo lib_system.ws_fs_server.
dfuwuid	O nome do atributo que receberá a string terminada por nulo que contém o ID da workunit DFU (DFUWUID) gerado para o job de monitoramento.
Return:	fMonitorLogicalFileName retorna uma string terminada com nulo contendo o ID da workunit DFU (DFUWUID).

A função **MonitorLogicalFileName** cria uma workunit de monitoramento de arquivos no servidor do DFU. Após ser recebida, o job entra no modo de monitoramento (que pode ser vista na tela da workunit DFU do eclwatch), que sonda em intervalos fixos (o padrão é 15 minutos). Se um arquivo com nome adequado chegar nesse intervalo, será acionado um *evento* com o nome do objeto acionador como subtipo do evento (consulte a função EVENT).

Esta função não permite caracteres curinga. Para monitorar arquivos físicos ou diretórios usando curingas, use a função MonitorFile .

Esse processo continua até que:

1) O número do eventos *shotcount* tenham sido gerados

2) O usuário aborte a workunit DFU.

As funções STD.File.AbortDfuWorkunit e STD.File.WaitDfuWorkunit podem ser usadas para abortar ou aguardar o job DFU passando por elas e retornando ao *dfuwuid*.

Observe as seguintes advertências e restrições:

1) Se existir um arquivo correspondente no momento em que o job do Monitorador DFU for iniciada, esse arquivo não gerará um evento. Ele só gerará o evento quando o arquivo tiver sido removido e recriado.

2) Se um arquivo for criado e depois removido (ou removido e depois recriado) entre os intervalos de sondagem, ele não será visto pelo monitor e não acionará um evento.

3) Os eventos são gerados apenas durante o intervalo de sondagem.

Exemplo:

```
EventName := 'MyFileEvent';
FileName  := 'test::myfile';
IF (STD.File.FileExists(FileName),
        STD.File.DeleteLogicalFile(FileName));
```

```
STD.File.MonitorLogicalFileName(EventName,FileName);
OUTPUT('File Created') : WHEN(EVENT(EventName,'*'),COUNT(1));

rec := RECORD
  STRING10 key;
  STRING10 val;
END;
afile := DATASET([{ 'A', '0'}], rec);
OUTPUT(afile,,FileName);
```