IBM Security Directory Integrator
Versão 7.2.0.1

Guia do Usuário

S517-9705-03

IBM Security Directory Integrator
Versão 7.2.0.1

Guia do Usuário

S517-9705-03

Nota

Antes de usar estas informações e o produto suportado por elas, leia as informações gerais no "Avisos" na página 241.

Aviso da Edição

Nota: Esta edição se aplica à versão 7.2.0.1 do programa licenciado *IBM Security Directory Integrator* **(5724-K74) a todas as liberações e modificações subsequentes, até que seja indicado de outra maneira em novas edições.**

Índice

Sobre esta Publicação

Esta publicação contém as informações requeridas para desenvolver soluções usando componentes que fazem parte do IBM® Security Directory Integrator.

Os componentes do IBM Security Directory Integrator foram projetados para administradores de rede que são responsáveis pela manutenção de diretórios do usuário e por outros recursos. Presume-se que você tenha experiência prática com a instalação e o uso tanto do IBM Security Directory Integrator quanto do IBM Security Directory Server.

As informações também são destinadas a usuários que são responsáveis pelo desenvolvimento, instalação e administração de soluções usando o IBM Security Directory Integrator. O leitor deve estar familiarizado com os conceitos e a administração de sistemas aos quais a solução desenvolvida seria conectada. Dependendo da solução, esses sistemas podem incluir, mas não estão limitados a, um ou mais dos produtos, sistemas e conceitos a seguir:

- IBM Security Directory Server
- IBM Security Identity Manager
- IBM Java™ Runtime Environment (JRE) ou Oracle Java Runtime Environment
- Microsoft Active Directory
- Sistemas operacionais Windows e UNIX
- Gerenciamento de segurança
- Protocolos da Internet, incluindo o Protocolo de Transporte de Hipertexto (HTTP), o Protocolo de Transporte de Hipertexto Seguro (HTTPS) e o Protocolo de Controle de Transmissões/Protocolo da Internet (TCP/IP)
- LDAP (Lightweight Directory Access Protocol) e serviços de diretório
- Um registro do usuário suportado
- Conceitos de autenticação e autorização
- SAP ABAP Application Server

Acesso a Publicações e Terminologia

Leia as descrições da biblioteca do IBM Security Directory Integrator Versão 7.2.0.1 e as publicações relacionadas que podem ser acessadas online.

Esta seção fornece:

- Uma lista de publicações na "Biblioteca do IBM Security Directory Integrator".
- Links para "Publicações Online" na página viii.
- Um link para "Website IBM Terminology" na página ix.

Biblioteca do IBM Security Directory Integrator

Os documentos a seguir estão disponíveis na biblioteca do IBM Security Directory Integrator:

- *IBM Security Directory IntegratorGuia de Administração do Versão 7.2.0.1 Federated Directory Server*

  Contém informações sobre o uso do console do Federated Directory Server para projetar, implementar e administrar soluções de integração de dados. Além

disso, contém informações sobre o uso do protocolo System for Cross-Domain Identity Management (SCIM) e da interface para gerenciamento de identidade.

- *IBM Security Directory Integrator Versão 7.2.0.1 Getting Started Guide*

  Contém um breve tutorial e uma introdução ao IBM Security Directory Integrator. Inclui exemplos para criar interação e aprendizado acessível do IBM Security Directory Integrator.

- *IBM Security Directory Integrator Versão 7.2.0.1 Users Guide*

  Contém informações sobre como utilizar o IBM Security Directory Integrator. Contém instruções para projetar soluções utilizando a ferramenta de designer do Security Directory Integrator (o Editor de Configuração) ou para executar as soluções prontas da linha de comandos. Também fornece informações sobre interfaces, conceitos e criação da AssemblyLine.

- *IBM Security Directory Integrator Versão 7.2.0.1 Installation and Administrator Guide*

  Inclui informações completas sobre instalação, migração a partir de uma versão anterior, configuração da funcionalidade de criação de log e o modelo de segurança subjacente da API do Servidor Remoto do IBM Security Directory Integrator. Contém informações sobre como implementar e gerenciar soluções.

- *IBM Security Directory Integrator Versão 7.2.0.1 Guia de Referência*

  Contém informações detalhadas sobre os componentes individuais do IBM Security Directory Integrator: Conectores, Componentes de Função, Analisadores, Objetos, etc. - os blocos de construção da AssemblyLine.

- *IBM Security Directory Integrator Versão 7.2.0.1 Problem Determination Guide*

  Fornece informações sobre as ferramentas, os recursos e as técnicas do IBM Security Directory Integrator que podem ajudar a identificar e resolver problemas.

- *IBM Security Directory Integrator Versão 7.2.0.1 Message Guide*

  Fornece uma lista de todas as mensagens informativas, de aviso e de erro associadas ao IBM Security Directory Integrator.

- *IBM Security Directory Integrator Versão 7.2.0.1 Password Synchronization Plug-ins Guide*

  Inclui informações completas para instalar e configurar cada um dos cinco Plug-ins de Sincronização de Senha da IBM: Windows Password Synchronizer, Sun Directory Server Password Synchronizer, IBM Security Directory Server Password Synchronizer, Domino Password Synchronizer e Password Synchronizer para UNIX e Linux. Também fornece instruções de configuração para o LDAP Password Store e o JMS Password Store.

- *IBM Security Directory Integrator Versão 7.2.0.1 Release Notes*

  Descreve os novos recursos e as informações mais recentes sobre o IBM Security Directory Integrator que não tenham sido incluídos na documentação.

Publicações Online

A IBM posta publicações de produto quando um produto é liberado e quando as publicações são atualizadas nos locais a seguir:

Biblioteca do IBM Security Directory Integrator
O site de documentação do produto (http://www-01.ibm.com/support/knowledgecenter/SSCQGF/welcome) exibe a página de boas-vindas e a navegação para essa biblioteca.

IBM Security Systems Documentation Central
O IBM Security Systems Documentation Central fornece uma lista

alfabética de todas as bibliotecas do produto e links dos Sistemas de Segurança da IBM para a documentação online para versões específicas de cada produto.

Centro de Publicações IBM
O site do Centro de Publicações IBM (http://www-05.ibm.com/e-business/linkweb/publications/servlet/pbi.wss) oferece funções de procura customizada para ajudá-lo a localizar todas as publicações IBM que você precisa.

Informações Relacionadas

Informações relacionadas ao IBM Security Directory Integrator estão disponíveis nos locais a seguir:

- O IBM Security Directory Integrator usa o cliente JNDI da Oracle. Para obter informações sobre o cliente JNDI, consulte *Java Naming and Directory Interface™ Specification* em http://download.oracle.com/javase/7/docs/technotes/guides/jndi/index.html .
- Informações que podem ajudar a responder suas perguntas relacionadas aos IBM Security Directory Integrator podem ser localizadas em https://www-947.ibm.com/support/entry/myportal/over-accesspubsview/software/security_systems/tivoli_directory_integrator.

Website IBM Terminology

O website IBM Terminology consolida a terminologia para bibliotecas do produto em um local. É possível acessar o website de Terminologia em http://www.ibm.com/software/globalization/terminology.

Acessibilidade

Os recursos de acessibilidade ajudam usuários com alguma deficiência física, como mobilidade restrita ou visão limitada, a usarem produtos de software com êxito. Com este produto, você pode utilizar tecnologias assistidas para escutar e navegar na interface. Também é possível usar o teclado no lugar do mouse para operar todos os recursos da interface gráfica com o usuário.

Para obter informações adicionais, consulte o Apêndice de Acessibilidade em *Configurando o Directory Integrator*.

Treinamento Técnico

Para obter informações sobre treinamento técnico, consulte o seguinte website do IBM Education em http://www.ibm.com/software/tivoli/education.

Informações de Suporte

O Suporte IBM fornece assistência a problemas relacionados a códigos e rotina, instalação de curta duração e questões de uso. Você pode acessar diretamente o site Suporte de Software IBM em http://www.ibm.com/software/support/probsub.html.

O *Resolução de problemas* fornece detalhes sobre:

- Quais informações coletar antes de contatar o Suporte IBM.
- Os vários métodos para contatar o Suporte IBM.

- Como usar o IBM Support Assistant.
- Instruções e recursos de determinação de problema para isolar e corrigir você mesmo o problema.

Declaração de Boas Práticas de Segurança

A segurança do sistema de TI envolve a proteção dos sistemas e de informações por meio de prevenção, detecção e resposta a acesso incorreto de dentro e fora da empresa. O acesso incorreto pode fazer com que informações sejam alteradas, destruídas, desapropriadas ou mal usadas ou pode causar danos ou mau uso de seus sistemas, incluindo para uso de ataques em outros. Nenhum sistema ou produto de TI deve ser considerado completamente seguro e um único produto, serviço ou medida de segurança não pode ser completamente efetivo como prevenção de uso ou acesso incorreto. Os sistemas, produtos e serviços IBM são projetados para fazerem parte de uma abordagem de segurança abrangente, que, necessariamente, envolverá procedimentos operacionais adicionais e poderá precisar de outros sistemas, produtos ou serviços para se tornar mais efetiva. A IBM NÃO GARANTE QUE QUAISQUER SISTEMAS, PRODUTOS OU SERVIÇOS SÃO IMUNES, OU DEIXARÃO SUA EMPRESA IMUNE DE CONDUTAS ILEGAIS OU MALICIOSAS DE QUALQUER PARTE.

Capítulo 1. Conceitos Gerais

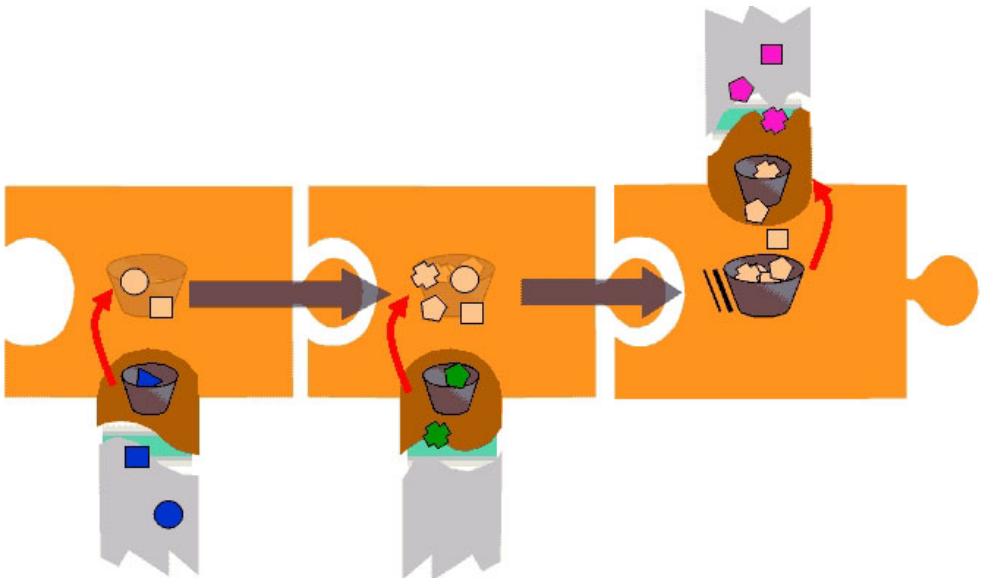
Esta seção introduz alguns dos conceitos básicos no IBM Security Directory Integrator, juntamente com os elementos da arquitetura que permitem construir suas soluções, suas características e seus comportamentos.

O AssemblyLine

Um AssemblyLine (AL) é um conjunto de componentes conectados para mover e transformar dados; ele descreve a "rota" ao longo da qual os dados serão passados. Os dados que são tratados através desse trajeto são representados como um objeto de *Entrada*. O AssemblyLine funciona com uma única entrada por vez em cada ciclo do AssemblyLine. Ele é a unidade de trabalho no IBM Security Directory Integrator e geralmente representa um fluxo de informações de uma ou mais origens de dados para um ou mais destinos.

Visão Geral

Alguns dos componentes que constituem o AssemblyLine recuperam dados de um ou mais *sistemas conectados*—digamos que os dados obtidos dessa maneira "alimentam" o AL. Os dados a serem processados são alimentados no AL, uma *Entrada* por vez, onde essas Entradas transportam Atributos com *valores* provenientes de entradas de diretório, linhas de banco de dados, e-mails, documentos do *Lotus Notes*, registros ou objetos de dados semelhantes. Cada entrada transporta os *Atributos* que contêm os valores de dados lidos a partir dos campos ou colunas no sistema de origem. Esses Atributos são renomeados, reformatados ou calculados como fluxos de processamento de um componente para o outro no AL. As novas informações podem ser "unidas" a partir de outras origens e todos os dados transformados, ou parte deles, podem ser gravados em armazenamentos de destino ou enviados para sistemas de destino, conforme desejado. Isso pode ser ilustrado assim:



Este diagrama representa a coleta de peças do grande jogo de quebra-cabeça como o AssemblyLine, os pontos e os quadrados em azul à esquerda no fluxo cinza

entrando de baixo como dados brutos de um fluxo de entrada e os bits na cor púrpura na parte superior direita como saída de dados em um fluxo de saída. O elemento na cor laranja mais escura cruzando uma peça do jogo de quebra-cabeça contendo o depósito indica um *Analisador*, transformando os dados brutos em dados estruturados, os quais podem iniciar o percurso no AssemblyLine (como elementos na cor mais clara em um depósito). A peça central do quebra-cabeça representa um Conector lendo dados já estruturados, por exemplo, de um banco de dados.

Os dados entram no AssemblyLine a partir de sistemas conectados usando "Conectores" na página 4 em algum tipo de *Modo* de entrada e saem posteriormente para um ou mais sistemas conectados usando Conectores em algum Modo de saída.

Os dados podem ser lidos a partir de sistemas orientados a registros, como um banco de dados ou uma fila de mensagens: neste caso, as diversas colunas na entrada são mapeadas imediatamente para os Atributos na Entrada de trabalho resultante, que é representada como um "depósito" na peça do quebra-cabeça à esquerda. Ou, os dados podem ser lidos a partir de um fluxo de dados, como um arquivo de texto em um sistema de arquivos, uma conexão de rede e outros. Neste caso, um Analisador pode ser prefixado para o Conector, para que faça sentido ao fluxo de entrada, e cortado em peças, após o qual ele pode ser designado a Atributos na Entrada de trabalho.

Assim que o primeiro Conector tiver concluído seu trabalho, o depósito de informações (a "Entrada de trabalho", denominada apropriadamente como *trabalho*) passa no AssemblyLine para o próximo Componente—na ilustração, um outro Conector. Como os dados do primeiro Conector estão disponíveis, agora eles podem ser usados como informações chave a serem recuperadas ou como dados de consulta no segundo sistema conectado. Assim que os dados relevantes são localizados, eles podem ser mesclados no *trabalho*, complementando os dados que ainda estão junto ao primeiro Conector.

Por último, os dados mesclados passam no AssemblyLine para a terceira peça do quebra-cabeça, ou Conector, desta vez no Modo de saída, que cuida da saída dos dados para o sistema conectado. Se o sistema conectado for orientado a registros, os diversos Atributos no *trabalho* serão apenas mapeados para colunas no registro; se o sistema conectado for orientado a fluxos, um Analisador poderá executar a formatação necessária.

Outros componentes, como "Componentes de Script" na página 23 e "Funções" na página 23, podem ser inseridos à vontade no AssemblyLine para executar operações nos dados no *trabalho*.

É importante lembrar sempre que o AssemblyLine é projetado e otimizado para trabalhar com um item de cada vez. Entretanto, se desejar fazer várias atualizações ou exclusões (por exemplo, processamento de mais de um único item por vez), grave os scripts do AssemblyLine. Se necessário, esse tipo de processamento pode ser implementado usando JavaScript, bibliotecas Java e funcionalidade padrão do IBM Security Directory Integrator, como agrupar em conjunto os dados em um armazém de dados classificado, por exemplo, com o Conector JDBC, em seguida, lê-lo novamente e processá-lo com um segundo AssemblyLine.

Os AssemblyLines são construídos, configurados e testados usando o IBM Security Directory Integrator Config Editor (CE), consulte o Capítulo 3, "O Configuration Editor", na página 77 para obter informações adicionais. O AssemblyLine possui

uma guia Fluxo de Dados no Config Editor. Esse é o local onde a lista de componentes que constituem o AL é mantida.

Todos os componentes em um AL são registrados automaticamente como variáveis de script. Portanto, se você tiver um Conector denominado ReadHRdump, ele e seus métodos poderão ser acessados diretamente do script usando a variável *ReadHRdump*. Como resultado, você desejará nomear seus componentes do AL do mesmo modo que as variáveis de script: Use apenas caracteres alfanuméricos, não inicie o nome com um número e não use caracteres nacionais especiais (por exemplo, å, ä), separadores (com exceção do sublinhado '_'), espaços em branco e assim por diante.

Há sempre um método alternativo para acessar o componente AL (por exemplo, a função `task.getConnector()`) mas uma convenção de nomenclatura consciente é sempre aconselhável.

Iniciar um AssemblyLine no IBM Security Directory Integrator é uma operação razoavelmente cara, pois envolve a criação de um novo encadeamento Java e geralmente a configuração de conexões com uma ou mais origens de dados. Considere com cuidado se seu design de solução pode ser criado para trabalhar com menos, em vez de mais, AssemblyLines distintos, em que cada AssemblyLine executa mais trabalho; por exemplo, usando Ramificações ou Comutadores para definir múltiplas operações tratadas por um único AL. Observe que cada operação pode, ainda, ser implementada como um AssemblyLine separado, mas elas podem ser integradas "ativas-e-prontas" a um único AL que efetua dispatch do trabalho para elas usando o Conector de AL ou a Função de AL. Isso também permite usar recursos como, Conjuntos de Conectores Globais, para gerenciar o uso de recursos e impulsionar o desempenho e a escalabilidade.

Componentes

Os AssemblyLines podem incluir os seguintes componentes:

- “Conectores” na página 4
- “Funções” na página 23
- “Componentes de Script” na página 23
- “AttributeMaps” na página 24
- “Componentes de Ramificação” na página 28

Além disso, os Conectores podem ter “Analisadores” na página 31 configurados; também, no nível de Sistema, de Configuração, de AssemblyLine, de mapa de Atributos e de Atributo, há opções para configurar o “Comportamento Nulo” na página 25.

Acessando Componentes do AL dentro do AssemblyLine

Cada componente do AL está disponível como uma variável pré-registrada do script, com o nome escolhido para o componente.

Observe que é possível carregar dinamicamente os componentes com chamadas em script para funções como `system.getConnector()`, embora isso não seja para usuários inexperientes.[1]

1. O objeto Conector obtido dessa chamada é um objeto da *Interface de Conector* e é a parte específica da origem de dados de um Conector AssemblyLine. Quando você altera o tipo de qualquer Conector, está realmente trocando sua inteligência de origem de

Transmissão dos Parâmetros do AssemblyLine

Há três formas de obter dados em um AssemblyLine:

- Gerando sua própria entrada inicial dentro do AssemblyLine; por exemplo, em um script de Prólogo.
- Alimentados a partir de um ou mais Agentes Iterativos[2].
- Iniciando o AssemblyLine com parâmetros de um outro AssemblyLine, usando o Conector AL ou o Componente de Função AL ou usando uma chamada de API.

Se você desejar iniciar um AssemblyLine com parâmetros de outro AssemblyLine, então terá duas opções:

- Usar o TCB (*Task Call Block*), que é o método mais indicado. Consulte "TCB (Task Call Block)" na página 66 para obter informações adicionais. Esta seção também discute técnicas para desativar e ativar dinamicamente componentes do AssemblyLine.
- Fornecer uma Entrada de Trabalho Inicial diretamente; consulte "Fornecendo uma IWE (Entrada de Trabalho Inicial)" na página 8 para obter detalhes.

  Nota: Essas opções são fornecidas para compatibilidade com versões anteriores.

Conectores

Os Conectores são usados para acessar e atualizar origens de informações. A tarefa de um Conector é nivelar o campo de reprodução para que você não tenha que lidar com os detalhes técnicos de trabalhar com vários armazéns de dados, sistemas, serviços ou transportes. Dessa forma, cada tipo de Conector está projetado para usar um protocolo ou API específico, tratando os detalhes do acesso à origem de dados para que você possa se concentrar nas relações e manipulações de dados, assim como customizar o processamento como controle de consistência e filtragem.

Visão Geral

Os conectores são usados para abstrair os detalhes de algum sistema ou armazenamento, fornecendo o mesmo conjunto de recursos de acesso. Isso permite trabalhar com um amplo intervalo de tecnologias e formatos diferentes de um modo consistente e previsível. Um AssemblyLine (AL) típico possui um Conector que fornece entrada e pelo menos um Conector que grava dados.

Há duas categorias de Conectores:

- A primeira categoria, em que o transporte e a estrutura do conteúdo de dados são conhecidos pelo Conector; ou seja, o esquema da origem de dados pode ser consultado ou detectado usando uma API bem conhecida, tal como JDBC ou LDAP.
- A segunda categoria, em que o mecanismo de transporte é conhecido, mas não a estrutura do conteúdo—que geralmente é semelhante a um fluxo de dados. Essa categoria requer um *Analisador* (consulte "Analisadores" na página 31 e a seção "Analisadores" na *Referência*) para interpretar ou gerar a estrutura de conteúdo para que o AssemblyLine funcione corretamente.

---

dados (a Interface de Conector) que fornece a funcionalidade para acessar dados em um sistema, serviço ou armazém de dados específico. A maior parte da funcionalidade de um Conector de AssemblyLine, incluindo os mapas de atributos, os Critérios de Link e os Ganchos, é fornecida pelo kernel e é mantida intacta quando você comuta os tipos de Conectores.

2. Um *Agente Iterativo* é uma notação abreviada de um Conector no Modo de Agente Iterativo.

Uma rica biblioteca de Conectores é uma das eficácias do IBM Security Directory Integrator. A lista de todos os Conectores inclusos no IBM Security Directory Integrator pode ser localizada no *Referência*. Mas você também pode escrever seu próprio conector em JavaScript ou até mesmo em Java; consulte "Implementing your own Components", *Referência*.

Cada Conector é projetado para um protocolo, API ou transporte específico e manipula a organização dos dados entre o tipo nativo do sistema conectado e os objetos Java. Diferente dos outros componentes, os Conectores possuem uma configuração de Modo que determina como esse Conector acessa seu sistema conectado; consulte “Modos do Conector” na página 6 para obter informações adicionais. Cada Conector suporta apenas um subconjunto dos modos apropriados para seu sistema conectado. Por exemplo, o File System Connector suporta somente um único modo de saída, AddOnly, e não Atualizar, Excluir ou CallReply. Ao usar um Conector, você deve primeiro consultar a documentação desse componente para obter uma lista dos modos suportados.

Nota: Os AssemblyLines podem consistir em vários, ou poucos, Conectores (e outros Componentes), conforme necessário para implementar seu Fluxo de Dados específico. Não há nenhuma limitação no sistema. Entretanto, o melhor a fazer é manter um AssemblyLine o mais simples possível para maximizar a capacidade de manutenção.

Ao selecionar um Conector para seu AssemblyLine, uma caixa de diálogo é exibida e possibilita que você escolha de qual tipo de Conector você deseja *herdar*. A herança é um conceito importante ao trabalhar com o IBM Security Directory Integrator, pois todos os componentes que você inclui nas soluções herdam algumas ou todas as suas características de um outro componente—de um dos tipos básicos ou de sua biblioteca de componentes pré-configurados: Conectores, Analisadores, Funções e outros, na seção Recursos de sua área de trabalho.

Quando usados em um AL, os Conectores fornecem uma opção Inicializar para controlar quando o componente é configurado; por exemplo, conexões feitas, recursos ligados e outros. Por padrão, todos os Conectores são inicializados quando o AssemblyLine é inicializado: a *Fase de Inicialização do AL*.

Os Conectores no modo Servidor ou Agente Iterativo alimentam o AssemblyLine e são responsáveis por alimentar o AL com uma nova Entrada de Trabalho para cada ciclo que o AL faz. A Entrada de Trabalho é passada de um componente para outro na seção Fluxo, seguindo qualquer lógica de Ramificação implementada, até o final do fluxo ser atingido. Neste ponto, o comportamento de fim do ciclo é iniciado, tal como o Agente Iterativo obtendo a próxima entrada de sua origem e passando-a para a seção Fluxo de um novo ciclo.

Enquanto um Agente Iterativo na seção Alimentações acionará realmente o Fluxo, um Agente Iterativo na seção Fluxo simplesmente obterá a próxima entrada e oferecerá seus Atributos de Dados para Mapeamento de Entrada na Entrada de Trabalho.

Nota: Você pode colocar um Conector no Modo de Agente Iterativo na seção Fluxo. Como tal, o Agente Iterativo funciona da mesma maneira que funcionaria em Alimentações: ele é inicializado, inclusive construindo seu conjunto de resultados com a chamada `selectEntries`, durante a inicialização do AL e recupera uma entrada (`getNextEntry`) em cada ciclo do AL. Entretanto, um Agente Iterativo na seção Fluxo não aciona o AL propriamente dito, como acionaria em Alimentações.

O comportamento da seção Alimentações é diferente para os modos Servidor e Agente Iterativo: Um Agente Iterativo espera ser o primeiro componente em execução no AL e somente lerá sua próxima entrada se a Entrada de Trabalho ainda não existir. Se o AL tiver passado por uma Entrada de Trabalho Inicial, os Agentes Iterativos não lerão quaisquer dados para esse primeiro ciclo. Isto significa também que os Agentes Iterativos são executados em uma série, com o segundo iniciando para retornar Entradas assim que o primeiro atingir o fim dos dados e não retornar nada (nulo).

Por outro lado, o Modo de Servidor faz com que o Conector ative um encadeamento de *listener* do servidor, por exemplo, para um retorno de chamada de porta IP ou de notificação de eventos e, então, passa o controle para o próximo Conector de Alimentações.

Quando você chamar um AL usando o AssemblyLine Function Component (AL FC), se usar o modo de ciclo manual, somente os componentes da seção Fluxo serão usados cada vez que o FC executar a chamada.

Há uma pasta **Conectores** em sua área de trabalho no Navegador do IBM Security Directory Integrator em que é possível manter a biblioteca de Conectores configurados. Esse também é o local no qual os Conjuntos de Conectores são definidos.

Modos do Conector

O modo de um Conector do AssemblyLine define a função que esse Conector representa no fluxo de dados e controla como o comportamento automatizado do AssemblyLine controla o Componente. Ele determina se deve ler de uma origem da entrada, gravar nela ou os dois.

Os conectores podem ser configurados para um destes modos padrão:

- Repetidor
- Consultar
- AddOnly
- Atualizar
- Excluir
- CallReply
- Servidor
- Delta

Esses modos são discutidos nas seções a seguir. Para obter uma descrição detalhada sobre o comportamento do modo Conector, bem como do comportamento do AssemblyLine em geral, consulte "Fluxogramas do AssemblyLine e dos Modos de Conectores", na publicação *Referência*.

Modo Repetidor:

Os Conectores no modo Repetidor são usados para varrer uma origem de dados e extrair seus dados. O Conector no modo Repetidor realmente repete por meio das entradas de origem de dados, lê seus valores de atributos e fornece cada entrada aos componentes da seção Fluxo do AssemblyLine, um de cada vez. Um Conector no modo Repetidor é geralmente conhecido como um Conector Repetidor ou apenas Repetidor.

Os AssemblyLines (exceto aqueles chamados com um IWE; consulte “Fornecendo uma IWE (Entrada de Trabalho Inicial)” na página 8) contêm geralmente pelo

menos um Conector no modo Repetidor. Os Repetidores fornecem o AssemblyLine com dados pela criação de Entradas de Trabalho e os transmitem para a seção Fluxo do AL.

Os componentes da seção Fluxo são carregados em ordem, iniciando no topo da lista Fluxo. Quando o processamento do Fluxo é concluído, o controle é transmitido novamente para o Repetidor, a fim de recuperar a próxima entrada.

Vários Repetidores em um AssemblyLine: Se você tiver mais de um Conector no modo Repetidor, esses Conectores serão empilhados na ordem em que aparecem na Configuração (e a Lista de Conectores no Editor de Configuração, na seção Alimentações) e serão processados um de cada vez. Então, se você estiver usando dois Repetidores, o primeiro lerá a partir de sua origem de dados, transmitindo a Entrada de Trabalho resultante ao primeiro que não for Repetidor, até que chegue ao fim do conjunto de dados. Quando o primeiro Repetidor chegar ao fim da sua origem de entrada, o segundo Repetidor começará a ler os dados.

Uma Entrada de Trabalho inicial é tratada como vinda de um Repetidor invisível processado antes de qualquer outro Repetidor. Isso significa que uma IWE é transmitida para o primeiro que não é Repetidor no AssemblyLine, ignorando todos os Repetidores durante o primeiro ciclo. Esse comportamento é visível na página Fluxo do AssemblyLine, no tópico "Fluxogramas do AssemblyLine e dos Modos de Conectores" da publicação *Referência*.

Suponha que você tenha um AssemblyLine com dois Repetidores, **a** precedendo **b**. O primeiro Repetidor, **a**, é usado até que **a** não retorne mais nenhuma entrada. Em seguida, o AssemblyLine alterna para **b**, ignorando **a**. Se uma IWE (Entrada de Trabalho Inicial) for transmitida para esse AssemblyLine, os dois Repetidores serão ignorados no primeiro ciclo, depois disso, o AssemblyLine começará a chamar **a**.

Algumas vezes, a IWE é usada para transferir parâmetros de configuração para um AssemblyLine, mas não dados. Entretanto, a presença de uma IWE faz com que os Repetidores no AssemblyLine sejam ignorados durante o primeiro ciclo. Se você não desejar que isso aconteça, deverá esvaziar o objeto Entrada de Trabalho chamando a função `task.setWork(null)` em um script Prólogo. Isso faz com que o primeiro Repetidor funcione normalmente.

Usando o Modo Repetidor:

É muito comum deixar que um Agente Iterativo acione o AssemblyLine.

O exemplo mais comum no uso de um Conector no modo Repetidor é:

1. Usando o Config Editor, inclua um Conector no modo Agente Iterativo em sua área de trabalho. Consulte o “Criando um Conector” na página 116.
2. Configure o modo (Agente Iterativo) e outros parâmetros de conexão para esse Conector na guia **Conexão**; os parâmetros necessários são marcados com um asterisco (*). Alguns Conectores requerem que você configure um Analisador como na sub-guia **Analisador**.
3. Configure o Mapa de Atributo; consulte “Mapeamento de Atributos de Entrada” na página 114.

Esses atributos mapeados são recuperados da origem de dados, colocados na entrada de *Trabalho* e passados para os Conectores na seção Fluxo no AssemblyLine.

Se você não criou o Conector diretamente dentro de um AssemblyLine, para usar esse Conector em um AssemblyLine, arraste o Conector de sue local em `<área_de_trabalho>/Resources/Connectors` para a seção **Alimentação** de um AssemblyLine.

Fornecendo uma IWE (Entrada de Trabalho Inicial): Essa é uma maneira alternativa de transmitir parâmetros usando um TCB e tem suporte quanto à compatibilidade com versões anteriores.

Quando um AssemblyLine for iniciado com a chamada `system.startAL()` a partir de um script, o AssemblyLine pode ainda ter os parâmetros transmitidos configurando os valores de atributo ou de propriedade da entrada Trabalho Inicial, que é acessada pela variável *work*. Ela é então a tarefa a ser aplicada nesses valores para definir os parâmetros do Conector; por exemplo, no Gancho **Prólogo – Inic** do AssemblyLine usando a função *connectorName*`.setParam()`.

Nota: Você deve limpar a Entrada de trabalho com a chamada `task.setWork(null)`; caso contrário, os Repetidores do AssemblyLine serão transmitidos no primeiro ciclo.

Você pode examinar o resultado do AssemblyLine, que é a **Entrada de Trabalho** quando o AssemblyLine for interrompido, usando a função `getResult()`. Consulte também "Conector fornecido pelo tempo de execução" em *Referência*.

Segue um exemplo de transmissão em um valor de parâmetro do Conector com uma IWE:

```
var entry = system.newEntry();
entry.setAttribute ("userNameForLookup", "John Doe");

// Here we start the AssemblyLine
var al = main.startAL ( "EmailLookupAL", entry );

// wait for al to finish
al.join();

var result = al.getResult();

// assume al sets the mail attribute in its working entry
task.logmsg ("Returned email = " + result.getString("mail"));
```

Modo Consultar:

O modo Consultar possibilita que você junte dados de diferentes origens de dados usando o relacionamento entre os atributos nestes sistemas. Um Conector no modo Consultar é referenciado com frequência como um Conector Consultar.

Para configurar um Conector de Consulta no Editor de Configuração, você deve indicar ao Conector como você define uma correspondência entre os dados que já estão no AssemblyLine e aqueles encontrados no sistema conectado. Isso é chamado de *Critérios de Link* do Conector e cada Conector de Consulta tem uma guia **Critérios de Link** associada em que você define as regras para localizar entradas correspondentes. Consulte “Critérios de Link” na página 20 para obter informações adicionais.

Usando o Modo Consultar: O exemplo mais comum no uso de um Conector no modo Consultar é:

1. Usando o Config Editor, inclua um Conector no modo Consultar em sua área de trabalho. Consulte o “Criando um Conector” na página 116.

2. Configure o modo (Consultar) e outros parâmetros de conexão para esse Conector na guia **Conexão**; os parâmetros necessários são marcados com um asterisco (*). Alguns Conectores requerem que você configure um Analisador como na sub-guia **Analisador**.
3. Configure o Mapa de Atributo; consulte "Mapeamento de Atributos de Entrada" na página 114.
4. Abra a guia **Critérios de Link** na janela de configuração do Conector e configure as regras para a correspondência de atributos. O resultado desse processo determinará quais entradas serão recuperadas do sistema conectado; aqui você tem várias opções:
   a. Clique em **Incluir** para incluir um novo Critério de Link e selecione um atributo do sistema conectado, o operador de correspondência (por exemplo, Igual, Começa Com e outros) e, em seguida, o atributo de Entrada de Trabalho a ser correspondido. Ao desempenhar a Consulta, o Conector cria a sintaxe de API ou protocolo subjacente com base nos Critérios de Link especificados, mantendo sua solução independente do tipo de sistema usado. É possível incluir vários Critérios de Link, que são conectados juntos pelo operador Booleano AND para construir a chamada de procura.
   b. Você também pode selecionar **Critérios de construção com script customizado**, que abre uma janela do editor de script na qual é possível criar sua própria cadeia de procura, passando-a volta ao Conector usando o objeto `ret.filter`. Por exemplo:

      ```
      ret.filter = "uid=" + work.getString("uid");
      ```

   Observe que as Expressões também pode ser usadas para especificar dinamicamente o Atributo ou Valor a ser usado para quaisquer Critérios de Link. Consulte "Expressões" na página 39 para obter informações. Consulte também "Critérios de Link" na página 20 para obter detalhes sobre Critérios de Link.

Os atributos lidos (e computados) no Mapa de Entrada estão disponíveis para outros Conectores de recebimento de dados e para a lógica de script que usa o objeto de entrada de Trabalho.

Se você não criou o Conector diretamente dentro de um AssemblyLine, então, para usar o Conector em um AssemblyLine, arraste-o de seu local em `<área_de_trabalho>/Resources/Connectors` para a seção **Fluxo** do AssemblyLine.

Modo AddOnly:

Os conectores em modo AddOnly, geralmente chamados de Conectores AddOnly, são usados para incluir novas entradas de dados em uma origem de dados.

Este modo de Conector não requer quase nenhuma configuração. Configure os parâmetros de conexão e selecione (mapeie) os atributos a serem gravados na Entrada de Trabalho.

Usando o Modo AddOnly: O exemplo mais simples e comum na usação de um Conector no modo AddOnly é:

1. Usando o Editor de Configuração, inclua um Conector no modo AddOnly em sua área de trabalho. Consulte o "Criando um Conector" na página 116.
2. Configure o modo (AddOnly) e outros parâmetros de conexão para esse Conector na guia **Conexão**; os parâmetros necessários são marcados com um asterisco (*). Alguns Conectores requerem que você configure um Analisador como na sub-guia **Analisador**.

3. Configure o Mapa de Atributo; consulte “Mapeamento do Atributo de Saída” na página 114.

Se você não criou o Conector diretamente dentro de um AssemblyLine, então, para usar o Conector em um AssemblyLine, arraste-o de seu local em `<área_de_trabalho>/Resources/Connectors` para a seção **Fluxo** do AssemblyLine.

Os atributos de entrada de trabalho que você mapeou são exibidos para o sistema conectado quando o Conector é chamado pelo AssemblyLine.

Modo Atualizar:

Os Conectores no modo Atualizar, geralmente referidos como Conectores Atualizar, são usados para incluir e modificar dados em uma origem de dados. Para cada entrada passada do AssemblyLine, o Conector Atualizar tenta localizar uma entrada correspondente a partir da origem de dados para ser modificada com os valores dos atributos de entrada recebidos. Se nenhuma correspondência for localizada, o Conector no modo Atualizar incluirá uma nova entrada.

Assim como os Conectores Procurar, você deve dizer ao Conector como você define uma correspondência entre dados que já estão no AssemblyLine e os que são encontrados no sistema conectado. Isso é chamado de Critérios de Link do Conector e cada Conector Atualizar possui uma guia Critérios de Link associada (consulte “Critérios de Link” na página 20) na qual você define as regras para localizar as entradas correspondentes. Se nenhuma entrada desse tipo for localizada, uma nova entrada será incluída na origem de dados. Entretanto, se uma entrada correspondente for localizada, ela será modificada. Se mais de uma entrada corresponder aos Critérios de Link, o Gancho Múltiplas Entradas Localizadas será chamado. Além disso, o Mapa de Saída pode ser configurado para especificar quais atributos serão usados durante uma operação Adicionar ou Modificar.

Quando executar uma operação Modificar, somente os atributos que estão marcados como Modificar (Mod) no Mapa de Saída são alterados na origem de dados. Se a entrada passada do AssemblyLine não tiver um valor para um atributo, o Comportamento Nulo para esse atributo se tornará significativo. Se configurado como Excluir, o atributo não existirá na entrada de modificação, portanto, o atributo não poderá ser alterado na origem de dados. Se estiver definido para *NULL*, o atributo existe na entrada de modificação, mas com um valor nulo, que significa que o atributo é excluído na origem de dados.

Um recurso importante que os Conectores Atualizar oferecem é a opção Computar Mudanças. Quando ativada, o Conector verifica primeiramente os novos valores contra os antigos e somente atualiza se e onde for necessário. Assim, é possível ignorar atualizações desnecessárias que podem ser valiosas se a operação de atualização for pesada para uma determinada origem de dados que você está atualizando.

Alguns Conectores Atualizar oferecem a opção para ignorar consultas desnecessárias ao executar Atualizações. Se o Conector suportar isso, você verá uma caixa de opção **Ignorar Consulta** próxima à caixa de opção **Computar Mudanças**. Quando selecionada, ela altera o comportamento do Conector de forma que nenhuma consulta seja executada para localizar uma entrada que corresponda aos critérios de link. Por essa razão, nenhum gancho Antes/Após Consulta é chamado. Além disso, o gancho Em Múltiplas Entradas não pode ser chamado. Com essa opção ativada, somente os critérios de procura são construídos e a

modificação é chamada diretamente. Isso é diferente do comportamento do conector padrão no modo Atualizar, pois se nenhuma entrada for localizada com os critérios de link, ela não será incluída como nova.

Usando o Modo Atualizar: O exemplo mais simples e comum no uso de um Conector no modo Atualizar é:

1. Usando o Config Editor, inclua um Conector no modo Atualizar em sua área de trabalho. Consulte o “Criando um Conector” na página 116.
2. Configure o modo (Atualizar) e outros parâmetros de conexão para esse Conector na guia **Conexão**; os parâmetros necessários são marcados com um asterisco (*). Alguns Conectores requerem que você configure um Analisador como na sub-guia **Analisador**.
3. Configure o Mapa de Atributo; consulte “Mapeamento do Atributo de Saída” na página 114.
4. Abra a guia **Critérios de Link** na janela de configuração do Conector e configure as regras para a correspondência de atributos. Aqui você tem duas escolhas:
   a. Clique em **Incluir** para incluir um novo Critério de Link e selecione um atributo do sistema conectado, o operador de correspondência (por exemplo, Igual, Começa Com e outros) e, em seguida, o atributo de entrada de Trabalho a ser correspondido. Ao desempenhar a Consulta, o Conector cria a sintaxe de API ou protocolo subjacente com base nos Critérios de Link especificados, mantendo sua solução independente do tipo de sistema usado. É possível incluir vários Critérios de Link, que são conectados juntos pelo operador Booleano AND para construir a chamada de procura.
   b. Você também pode selecionar **Critérios de construção com script customizado**, que abre uma janela do editor de script na qual é possível criar sua própria cadeia de procura, passando-a volta ao Conector usando o objeto `ret.filter`. Por exemplo:

      ```
      ret.filter = "uid=" + work.getString("uid");
      ```

   Observe que as Expressões também pode ser usadas para especificar dinamicamente o Atributo ou Valor a ser usado para quaisquer Critérios de Link. Consulte “Expressões” na página 39 para obter informações. Consulte também “Critérios de Link” na página 20 para obter detalhes sobre Critérios de Link.

As entradas com os Atributos selecionados para serem mapeados na entrada são incluídas na origem de dados durante a execução do AssemblyLine.

Para usar o Conector em um AssemblyLine, arraste-o de seu local em `<área_de_trabalho>/Resources/Connectors` para a seção **Fluxo** de um AssemblyLine. Agora você pode mapear atributos de entrada de Trabalho para a saída, arrastando os atributos que foram mapeados anteriormente para o Conector Atualizar na janela **Mapas de Atributos** do AssemblyLine.

Você também pode criar Atributos totalmente novos, clicando com o botão direito do mouse no Conector nessa janela e selecionando **Incluir Item do Mapa de Atributos**.

Nota: No modo Atualizar, várias entradas podem ser atualizadas. Consulte "Fluxogramas do AssemblyLine e dos Modos de Conectores", na publicação *Referência*.

Modo Excluir:

Conectores no modo Excluir, geralmente chamados de Conectores de Exclusão são usados para excluir dados de uma origem de dados.

Para cada entrada transmitida ao Conector de Exclusão, ele tenta localizar dados correspondentes no sistema conectado. Se uma única entrada correspondente for encontrada, ela será excluída; caso contrário, o gancho Em Nenhuma Correspondência será chamado se nenhuma correspondência for encontrada ou o gancho Em Múltiplas Entradas se mais de uma única correspondência for encontrada. Como nos modos Consulta e Atualizar, o modo Excluir exige que você defina regras para encontrar a entrada correspondente para exclusão. Isso é configurado na guia Critérios de Link do Conector.

Alguns Conectores de Exclusão oferecem a opção de ignorar consultas desnecessárias ao fazer Exclusões. Se o Conector suportá-la, você verá uma caixa de opção Ignorar Consulta ao lado da caixa de opção Computar Alterações. Quando selecionada, ela altera o comportamento do Conector de forma que nenhuma consulta seja executada para localizar uma entrada que corresponda aos critérios de link. Por essa razão, nenhum gancho Antes/Após Consulta é chamado. Além disso, o gancho Em Múltiplas Entradas não pode ser chamado. Com essa opção ativada, apenas os critérios de procura são criados ea exclusão é chamada diretamente.

Usando o Modo Excluir: O exemplo mais comum e simples na usação de um Conector no modo Excluir é:

1. Usando o Config Editor, inclua um Conector no modo Excluir em sua área de trabalho. Consulte o "Criando um Conector" na página 116.
2. Configure o modo (Excluir) e outros parâmetros de conexão para esse Conector na guia **Conexão**; os parâmetros necessários são marcados com um asterisco (*). Alguns Conectores requerem que você configure um Analisador como na sub-guia **Analisador**.
3. Configure o Mapa de Atributo; consulte "Mapeamento de Atributos de Entrada" na página 114.
4. Na guia Mapa de Atributos de Entrada, você seleciona os atributos na lista Atributo do Conector e, em seguida, arrasta-os para seu Mapa de Entrada. Também é possível incluir e remover atributos manualmente.

   Nota: O Mapa de Entrada é usado no modo Excluir para ler a entrada correspondente localizada na origem de dados para o objeto de entrada *conn*, que pode então ser usado em seus scripts (por exemplo, para determinar se a entrada deverá ser realmente excluída).
5. Abra a guia **Critérios de Link** na janela de configuração do Conector e configure as regras para a correspondência de atributos. O resultado desse processo determinará quais entradas serão recuperadas do sistema conectado; aqui você tem várias opções:
   a. Clique em **Incluir** para incluir um novo Critério de Link e selecione um atributo do sistema conectado, o operador de correspondência (por exemplo, Igual, Começa Com e outros) e, em seguida, o atributo de Entrada de Trabalho a ser correspondido. Ao desempenhar a Consulta, o Conector cria a sintaxe de API ou protocolo subjacente com base nos Critérios de Link especificados, mantendo sua solução independente do tipo de sistema usado. É possível incluir vários Critérios de Link, que são conectados juntos pelo operador Booleano AND para construir a chamada de procura.

b. Você também pode selecionar **Critérios de construção com script customizado**, que abre uma janela do editor de script na qual é possível criar sua própria cadeia de procura, passando-a volta ao Conector usando o objeto `ret.filter`. Por exemplo:

```
ret.filter = "uid=" + work.getString("uid");
```

Consulte “Critérios de Link” na página 20 para obter informações adicionais sobre os Critérios de Link.

Se você não criou o Conector diretamente dentro de um AssemblyLine, então, para usar o Conector em um AssemblyLine, arraste-o de seu local em `<área_de_trabalho>/Resources/Connectors` para a seção **Fluxo** do AssemblyLine.

Modo CallReply:

O modo CallReply é usado para fazer solicitações para serviços de origem de dados, como serviços da Web, que exigem o envio de parâmetros de entrada e recebem uma resposta com valores retornados.

Ao contrário de outros modos, o modo CallReply oferece acesso aos Mapas de Atributos de Entrada e de Saída.

O Conector primeiro executa uma operação no mapa de Saída e com isso chama um sistema externo, com parâmetros fornecidos pela operação do Mapa de saída. Isso é imediatamente seguido por uma operação do mapa de Entrada, além de coletar a resposta do sistema externo.

Usando o modo CallReply: O exemplo mais simples e comum na usação de um Conector no modo CallReply é:

1. Usando o Config Editor, inclua um Conector no modo CallReply em sua área de trabalho. Consulte o “Criando um Conector” na página 116. Poucos Conectores suportam esse modo.
2. Configure o modo (CallReply) e outros parâmetros de conexão para esse Conector na guia **Conexão**; os parâmetros necessários são marcados com um asterisco (*). Alguns Conectores requerem que você configure um Analisador como na sub-guia **Analisador**.
3. Configure o Mapa de Atributos de Saída; consulte “Mapeamento do Atributo de Saída” na página 114.
4. Configure o Mapa de Atributos de Entrada; consulte “Mapeamento de Atributos de Entrada” na página 114.

Tenha em mente que os Atributos do Mapa de Saída são fornecidos ao sistema conectado como parâmetros de entrada; os Atributos mapeados por meio do mapa de Atributos de Entrada contêm a resposta do sistema conectado.

Modo Servidor:

O modo Servidor, disponível em um número selecionado de Conectores, está projetado para fornecer a funcionalidade para esperar um evento recebido, efetuar dispatch de um encadeamento lidando com o evento e enviar uma resposta de volta para o originador.

Atualmente, todos os conectores no Modo Servidor são baseados em conexão. Como resultado, qualquer AssemblyLine (AL) que usar um Conector no Modo Servidor em sua seção Alimentações será inicializado e, então, aguardará uma conexão recebida (por meio de uma conexão TCP, HTTP, LDAP, Serviços da Web,

SNMP); quando uma conexão for iniciada, o conector no Modo Servidor clonará o AL do qual ele faz parte e retomará a espera pelo próximo evento (ou seja, uma nova inicialização de conexão). Enquanto isso, no AL trabalhador clonado, o Conector no Modo Servidor se coloca no modo Agente Iterativo e inicia a leitura de dados da conexão. Os dados obtidos da conexão são, então, alimentados para o restante do AL no modo Agente Iterativo normal, incluindo o acompanhamento do fluxo padrão de Ganchos do Agente Iterativo, a leitura das entradas de eventos, uma por vez, e a transmissão dessas entradas aos outros componentes de Fluxo para processamento até o esgotamento dos dados para leitura. No final de cada ciclo (geralmente haverá somente um), o AL conduzido pelo conector no Modo Servidor enviará uma resposta de volta ao cliente, a menos que você decida ignorar a fase de resposta com, por exemplo, `system.skipEntry();`.

Assim que o AL alimentado estiver concluído (ou seja, a origem de dados estiver esgotada) esse encadeamento será finalizado; nesse momento, o AL trabalhador será limpo e, se necessário, o Gerenciador de Conjunto será informado que essa instância do AL está disponível novamente.

Enquanto isso, o conector no Modo Servidor original ainda está atendendo ativamente a mais inicializações de conexão.

Nota: Em determinadas condições raras, como quando você emite mais de 5 pedidos do cliente para o servidor em paralelo, principalmente no sistema operacional zOS, os clientes SNMP saem, apresentando exceções de Protocol Data Unit (PDU) inválidas. Entretanto, em uma situação mais realista da vida real, um agente como o Conector de Servidor SNMP é raramente consultado intensamente por múltiplos gerenciadores, como o Conector SNMP.

O Conector SNMP possui um parâmetro de configuração não documentado nomeado *snmpWalkTimeout*. Você pode substituir o padrão para esse parâmetro, que é 5000 ms. o parâmetro não é acessível usando o Config Editor. Você pode configurar o valor de substituição para esse parâmetro utilizando JavaScript. Configure o valor desejado para o parâmetro snmpWalkTimeout no seguinte formato:

```
thisConnector.connector.setParam("snmpWalkTimeout", "100000");
```

O Modo Servidor e o ALPool: O processo de criação de um AL de clone pode ser otimizado usando o AssemblyLine Pool (ALPool). Na detecção de eventos, o Conector no modo Servidor prossegue com a seção Fluxo desse AL ou, se um ALPool estiver configurado para esse AL, ele entra em contato com o processo do Gerenciador de Conjuntos para solicitar uma instância do AL disponível para manipular esse evento.

Quando um AssemblyLine com um Conector no modo Servidor usar o ALPool, o ALPool executará instâncias do AL do início ao fim. Antes da instância do AL no ALPool fechar os Conectores de Fluxo, o ALPool recuperará esses Conectores em um conector em conjunto configurado que será reutilizado na próxima instância do AL criada pelo ALPool. Basicamente, o ALPool usa o método `tcb.setRuntimeConnector()`.

Há duas propriedades de sistema que controlam o comportamento do conjunto de Conectores:

com.ibm.di.server.connectorpooltimeout
: Essa propriedade define o tempo limite em segundos antes da liberação de um conjunto de conectores agrupados.

Tabela 1. Tabela de Valores da Propriedade Tempo Limite do Conjunto de Conectores

Valor	Significado
< 0	Desativar conjunto de Conectores
0	Tempo limite desativado; os Conectores do conjunto nunca expiram
> 0	Número de segundos antes dos Conectores em conjunto expirarem

com.ibm.di.server.connectorpoolexclude
Esta propriedade define os tipos de Conectores que são excluídos do conjunto. Se o nome da classe de um Conector aparecer nessa lista separada por vírgulas, ele não será incluído no conjunto de Conectores.

Quando uma nova instância do AL (AssemblyLine) for criada pelo ALPool, ele procurará um conjunto de conectores agrupados disponível que, se presente, será fornecido a essa nova Instância do AL como conectores fornecidos em tempo de execução. Isso assegura o fluxo adequado do AL em geral, em termos de Mapeamento de Atributo, execução de Gancho e outros.

Os Conectores nunca são compartilhados. Eles são apenas designados a uma única instância do AL ao serem usados.

Usando o Modo Servidor: O padrão mais comum para usar um Conector no Modo Servidor é:

1. Usando o Config Editor, inclua um Conector no modo Servidor em sua área de trabalho. Consulte o "Criando um Conector" na página 116.
2. Configure o modo (Servidor) e outros parâmetros de conexão para esse Conector na guia **Conexão**; os parâmetros necessários são marcados com um asterisco (*). Alguns Conectores requerem que você configure um Analisador como na sub-guia **Analisador**.
3. Configure o Mapa de Atributo; consulte "Mapeamento de Atributos de Entrada" na página 114.

Esses atributos mapeados são recuperados da origem de dados, colocados na entrada de *Trabalho* e passados para os Conectores na seção Fluxo no AssemblyLine.

Nota:

1. Os conectores no modo Servidor são especiais, pois geralmente precisam retornar algumas informações para o cliente ao qual estão conectados. Devido a essa natureza, em muitos casos, a configuração de Mapas de Atributos clicando em **Descobrir Atributos** não resultará em qualquer Esquema significativo; portanto, na maioria dos casos, você precisará configurar mapas de atributos de forma totalmente manual, selecionando **Incluir Novo Atributo**; alternativamente, exclua aqueles desnecessários, selecionando um mapeamento e excluindo-o. Os dados mapeados dessa maneira são enviados de volta ao cliente, para cada ciclo do AL; apesar de que conforme mencionado antes, em um modo de pedido ou resposta típico de operação, geralmente haverá somente um ciclo.
2. Os Conectores no Modo Servidor com base nos protocolos TCP possuem um parâmetro chamado Lista Não-processada de Conexão que controla o comprimento da fila para conexões recebidas.
3. Na lista de componentes do AssemblyLine, você pode observar as seções chamadas Alimentações e Fluxo. Os Conectores no Modo Servidor observam uma terceira fase no processamento do AssemblyLine, a fase de *Resposta*. O Mapa de Saída e os Ganchos Resposta fazem parte do próprio Conector no

Modo Servidor na tela, mas a execução do comportamento de resposta é feita após o processamento da seção Fluxo ser concluído.

Observe que uma chamada `system.skipEntry();` instruirá o AssemblyLine a ignorar o comportamento de resposta, e como tal, nenhuma resposta será feita por um Conector no Modo Servidor. Se você preferir simplesmente ignorar os componentes restantes da seção Fluxo e, contudo, enviar a resposta, use em seu lugar a chamada `system.exitBranch("Flow");`.

Se você não criou o Conector diretamente dentro de um AssemblyLine, para usar esse Conector em um AssemblyLine, arraste o Conector de sue local em `<área_de_trabalho>/Resources/Connectors` para a seção **Alimentação** de um AssemblyLine.

Modo Delta:

O modo Delta destina-se a simplificar o aplicativo de alterações a dados, fornecendo modificações incrementais ao sistema conectado, baseado em códigos de operação delta.

Os códigos de operação delta são configurados pelo recurso de mecanismo Delta do Agente Iterativo (guia **Delta** para Agentes Iterativos) ou por Conectores de Detecção de Mudanças, como os Conectores IBM Security Directory Server, LDAP ou Active Directory (AD), ou aqueles para Mudanças do RDBMS e do Lotus/Domino; ou analisando essas informações delta com os Analisadores Lightweight Directory Interchange Format (LDIF) ou Directory Services Markup Language (DSML).

Em versões anteriores à IBM Security Directory Integrator V6.1, as capturas instantâneas gravadas no Armazenamento Delta, um recurso do Armazenamento do Sistema, durante o processamento do mecanismo Delta foram consolidadas imediatamente. Como resultado, o mecanismo Delta consideraria uma entrada alterada como manipulada, mesmo que o processamento da seção Fluxo do AL falhasse. Essa limitação é determinada pelo parâmetro *Commit* na guia Delta do Conector. A configuração desse parâmetro controla quando o mecanismo Delta consolida capturas instantâneas obtidas de dados recebidos no Armazenamento do Sistema.

O modo Delta está disponível apenas para os Conectores LDAP e JDBC.

Nota: Um Conector no modo Delta deve ser unido a outro Conector que fornece informações Delta; caso contrário, o modo Delta não tem nenhum código de operação delta com o qual trabalhar.

Os recursos Delta do IBM Security Directory Integrator (consulte Capítulo 7, “Deltas”, na página 209) destinam-se a facilitar as soluções de sincronização. É possível considerar os recursos Delta do sistema como divididos em duas seções: *Detecção* e *Aplicação*.

Detecção Delta: O IBM Security Directory Integrator fornece vários mecanismos e ferramentas de detecção de alterações ou delta:

Mecanismo Delta

Esse recurso está disponível para Conectores no modo Repetidor. Se ativado na guia Delta do Repetidor, o recurso do mecanismo Delta usa o Armazenamento do Sistema para obter uma captura instantânea de dados que estão sendo repetidos. Depois de sucessivas execuções, cada entrada

repetida é comparada ao banco de dados da captura instantânea (o Armazenamento Delta) para ver o que foi alterado.

Conector de Detecção de Alterações

Esses componentes alavancam informações do sistema conectado para detectar alterações e são usados no Modo Repetidor ou Servidor, dependendo do Conector. Por exemplo, o modo Agente Iterativo é usado para vários Conectores de Detecção de Mudanças, como aqueles para LDAP, Log de Mudanças do IBM Security Directory Server, bem como Conectores de Detecção de Mudanças do RDBMS, Active Directory e Notes/Domino.

Os Conectores de Detecção de Alterações foram designados para se comportar de uma maneira comum, bem como para fornecer os mesmos rótulos de parâmetros para configurações comuns. Os Conectores são:

- Log de Mudanças do IBM Security Directory Server
- Detecção de Alterações do AD (Active Directory)
- Detecção de Alterações do Domino
- Detecção de Alterações do Sun Directory (openLDAP, SunOne, iPlanet, etc.)
- Detecção de Alterações do RDBMS (DB2, Oracle, SQL server, etc.)
- Log de Mudanças do z/OS LDAP

Nota: O sistema operacional z/OS não é suportado a partir do IBM Security Directory Integrator Versão 7.2.

Consulte a seção "Conectores" na *Referência* para obter informações adicionais sobre esses Conectores.

O recurso do mecanismo Delta relata alterações inteiramente específicas para os valores individuais de atributos. Esse grau refinado de detecção de alterações também está disponível durante a análise de arquivos LDIF. Outros componentes estão limitados a simplesmente relatar se uma entrada inteira for incluída, modificada ou excluída.

Selecionando a caixa de opção **Permitir Chaves Delta Duplicadas** na guia Delta do Repetidor, você indica que permite chaves delta duplicadas, para aqueles casos de AssemblyLines de execução longa que precisam processar as mesmas entradas mais de uma vez. Isso significa que entradas duplicadas podem ser tratadas nos AssemblyLines que usam Log de Mudanças ou Conectores de Detecção de Alterações, conectores no modo Delta e armazenamentos no modo Delta, quando uma entrada já tiver sido atualizada.

Atenção: Há uma possibilidade de ter, por exemplo, um AssemblyLine com vários Logs de Mudanças e Conectores no modo Delta. Nesse caso, se o Conector no modo Delta estiver apontando para o mesmo sistema subjacente que o Conector do Log de Mudanças, a operação Delta poderia ativar o Log de Mudanças novamente. Como não há uma maneira de diferenciar entre alterações recém-recebidas e aquelas ativadas pelo mecanismo Delta, você deverá considerar cuidadosamente o seu cenário a fim de não entrar em um loop sem fim.

As informações delta computadas pelo mecanismo Delta são armazenadas no objeto Entrada de Trabalho e, dependendo do componente ou do recurso de Detecção de Alterações usado, podem ser armazenadas como um *Código de operação em Nível de Entrada*, no *Nível de Atributo* ou até mesmo no *Valor-Nível de Atributo*.

Como exemplo, configure um Conector do Sistema de Arquivos com o recurso do mecanismo Delta ativado. Faça com que ele seja repetido em um documento XML simples que possa ser facilmente modificado em um editor de texto. Por exemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<DocRoot>
    <Entry>
        <Telephone>
            <ValueTag>111-1111</ValueTag>
            <ValueTag>222-2222</ValueTag>
            <ValueTag>333-3333</ValueTag>
        </Telephone>
        <Birthdate>1958-12-24</Birthdate>
        <Title>Full-Time SDI Specialist</Title>
        <uid>jdoe</uid>
        <FullName>John Doe</FullName>
    </Entry>
</DocRoot>
```

Assegure-se de usar o caractere especial de mapa de atributo para mapear tudo, o asterisco (*). Esse é o único Atributo que você precisa em seu mapa para garantir que todos os Atributos retornados sejam mapeados no objeto Entrada de trabalho.

Agora, inclua um Componente de Script com o seguinte código:

```
// Get the names of all Attributes in work as a String array
var attName = work.getAttributeNames();
// Print the Entry-level delta op code
task.logmsg(" Entry (" +
    work.getString( "FullName" ) + ") : " +
    work.getOperation() );
// Loop through all the Attributes in work
for (i = 0; i < attName.length; i++) {
  // Grab an Attribute and print the Attribute-level op code
 att = work.getAttribute( attName[ i ] );
 task.logmsg("    Att ( " + attName[i] + ") : " + att.getOperation()
);
  // Now loop through all the Attribute's values and print their op
codes
 for (j = 0; j < att.size(); j++) {
  task.logmsg( "      Val (" +
                     att.getValue( j ) + ") : " +
                     att.getValueOperation( j ) );
 }
}
```

Na primeira vez que você executar esse AL, o código do Componente de Script criará esta saída de log:

```
12:46:31   Entry
(John Doe) : add
12:46:31      Att ( Telephone) : replace
12:46:31        Val (111-1111) :
12:46:31        Val (222-2222) :
12:46:31        Val (333-3333) :
12:46:31      Att ( Birthdate) : replace
12:46:31        Val (1958-12-24) :
12:46:31      Att ( Title) : replace
12:46:31        Val (Full-Time SDI Specialist) :
12:46:31      Att ( uid) : replace
12:46:31        Val (jdoe) :
12:46:31      Att ( FullName) : replace
12:46:31        Val (John Doe) :
```

Como essa entrada não foi encontrada no Armazenamento Delta vazio anteriormente, ele é marcado no nível da entrada como *new*. Além disso, cada um

de seus Atributos possui um código *replace*, significando que todos os valores foram alterados (o que faz sentido, já que o Delta está informando que esses dados são novos).

Faça as seguintes alterações em seu arquivo XML:

1. Altere o último valor do número de Telephone para 333-3334.
2. Exclua Birthdate.
3. Inclua um novo Atributo Address.

Sua Configuração resultante deve ser semelhante a esta:

```
<?xml version="1.0" encoding="UTF-8"?>
<DocRoot>
    <Entry>
        <Telephone>
            <ValueTag>111-1111</ValueTag>
            <ValueTag>222-2222</ValueTag>
            <ValueTag>333-3334</ValueTag>
        </Telephone>
        <Title>Full-Time SDI Specialist</Title>
        <uid>jdoe</uid>
        <FullName>John Doe</FullName>
        <Address>123 Willowby Lane</Address>
    </Entry>
</DocRoot>
```

Execute novamente o AL. Dessa vez, a saída de log terá a seguinte aparência:

```
13:53:22   Entry (John Doe) : modify
13:53:22      Att ( Telephone) : modify
13:53:22        Val (111-1111) : unchanged
13:53:22        Val (222-2222) : unchanged
13:53:22        Val (333-3334) : add
13:53:22        Val (333-3333) : delete
13:53:22      Att ( Birthdate) : delete
13:53:22        Val (1958-12-24) : delete
13:53:22      Att ( uid) : unchanged
13:53:22        Val (jdoe) : unchanged
13:53:22      Att ( Title) : unchanged
13:53:22        Val (Full-Time SDI Specialist) : unchanged
13:53:22      Att ( Address) : add
13:53:22        Val (123 Willowby Lane) : add
13:53:22      Att ( FullName) : unchanged
13:53:22        Val (John Doe) : unchanged
```

Agora, a entrada é marcada como *modify* e os Atributos refletem as modificações em cada um deles. Como você pode ver, o Atributo Birthdate é marcado como *delete* e Address como *add*. Foi por isso que você utilizou o caractere especial de mapeamento completo para o Mapa de Entrada. Se você tivesse mapeado apenas os Atributos que existiam na primeira versão desse documento XML, não teríamos recuperado Address quando ele apareceu na entrada.

Observe especialmente as duas últimas entradas de valor no Atributo Telephone, marcadas como *modify*. A alteração em um dos valores desse Atributo resultou em dois itens Delta: um valor *delete* e, em seguida, um *add*.

Para construir um AssemblyLine de sincronização de dados em versões anteriores do IBM Security Directory Integrator, você tinha que criar um script para tratar o controle de fluxo. Embora pudéssemos receber *adds*, *modifies* e *deletes* do componente ou recurso de alteração, um Conector apenas poderia ser configurado como um dos dois modos de saída necessários: Atualizar ou Excluir. Portanto, você tinha dois Conectores apontando para o mesmo sistema de destino e colocava o

script na opção Antes de Executar o Gancho de cada um para ignorar a entrada se seu código de operação não correspondesse ao modo deste componente; ou podia ter um único Conector (modo Atualizar ou Excluir) no estado *Passivo* e, em seguida, controlar sua execução a partir do código do script em que você verificou o código de operação. Isso ainda significa que embora você soubesse o que tinha sido alterado no caso de uma entrada modificada, o seu Conector no Modo Atualizar ainda leria os dados originais antes de gravar as alterações novamente na origem de dados. Isso pode levar a tráfego de rede ou de origem de dados indesejado quando você estiver apenas alterando um único valor em um Atributo relacionado a um grupo com diversos valores que contenha milhares de valores.

Entre no modo *Delta* de Conectores.

Aplicação Delta (Modo Delta de Conectores): O modo Delta foi projetado para simplificar a aplicação de informações delta; ou seja, fazer as alterações reais de várias maneiras.

Em primeiro lugar, o modo Delta manipula todos os tipos de deltas: inclusões, modificações e exclusões. Isso reduz o número de ALs de sincronização de dados a dois Conectores: um Conector de Detecção Delta na seção Alimentações (Feeds), para recolher as alterações, e um segundo Conector no modo Delta, para aplicar essas alterações a um sistema de destino.

Além disso, o modo Delta aplicará as informações delta no nível mais inferior suportado pelo próprio sistema de destino. Isso é feito verificando primeiro a Interface de Conector para detectar que nível de modificação incremental é suportado pela origem de dados[3]. Se você estiver trabalhando com um diretório LDAP, o modo Delta realizará inclusões e exclusões de valores de Atributos. No contexto de um RDBMS tradicional (JDBC), operações subsequentes de exclusão e, em seguida, de inclusão de um valor de coluna não fazem sentido e, por isso, são manipuladas como uma substituição de valores para esse Atributo.

Isso é tratado automaticamente pelo modo Delta para aquelas origens de dados que suportam essa funcionalidade[4]. Se a origem de dados oferecer chamadas otimizadas para manipular modificações incrementais e essas modificações forem suportadas pela Interface de Conector, o modo Delta usará as chamadas otimizadas. Por outro lado, se o sistema conectado não oferecer mecanismos de atualização delta "inteligentes", o modo Delta simulará esses mecanismos da melhor maneira possível, realizando consultas pré-atualização (como no modo Atualizar), cálculos de alterações e a aplicação subsequente das alterações detectadas.

Critérios de Link:

Os Critérios de Link são usados para indicar a um Conector nos modos de Atualização, Consulta e Exclusão como você define uma correspondência entre atributos de dados no AssemblyLine e aqueles localizados no sistema conectado.

Os Critérios de Link são acessíveis no Editor de Configuração por meio da guia **Critérios de Link**, que é ativada apenas nos modos de Conector Atualizar, Consulta e Excluir.

3. Observe que os únicos Conectores que suportam as modificações incrementais são o LDAP e o JDBC, pois os diretórios LDAP fornecem essa funcionalidade

4. Além disso, você pode controlar esses comportamento interno por meio de parâmetros de configuração e códigos de Gancho.

Há dois tipos de Critérios de Link, **Simples** e **Avançado**.

Critérios de Link Simples: Para cada Critério de Link simples, especifique o Atributo de Conector (os atributos definidos no Esquema de Conector), o Operador a ser usado (por exemplo, Contém, Igual e outros) e o Valor a ser usado com a operação. O valor usado pode ser inserido diretamente, ou pode referir-se ao valor de um atributo na Entrada de Trabalho que está disponível neste ponto no fluxo do AssemblyLine. Quando o Conector executa a operação Procurar (para os modos Procurar, Atualizar e Excluir), ele converte os Critérios de Link para a chamada específica da origem de dados, possibilitando que você mantenha sua solução independente da tecnologia essencial.

Se você desejar construir um Critério de Link usando o valor de um atributo na Entrada de Trabalho, simplesmente use o nome do atributo no campo Valor dos Critérios de Link, precedido pelo cifrão ($). Portanto, se você desejar corresponder o atributo nomeado *cn* com um atributo na Entrada de Trabalho denominado FullName, seu Critério de Link será especificado como:

```
cn EQUALS $FullName
```

Se quiser localizar diretamente uma determinada pessoa, defina os Critérios de Link com um valor de constante literal:

```
cn EQUALS Joe Smith
```

Nota: O cifrão ($) corresponde ao primeiro valor de um atributo com vários valores somente. Se você desejar corresponder um atributo na origem de dados com qualquer um dos múltiplos valores armazenados em um atributo de Entrada de Trabalho, use o símbolo *arroba* (@). Por exemplo:

```
dn EQUALS @members
```

Este exemplo tenta corresponder o atributo dn no sistema conectado com qualquer um dos valores do atributo de multivalores na Entrada de Trabalho nomeada *members*.

Um Conector pode ter múltiplos Critérios de Link definidos, e eles são normalmente conectados juntos, pelo uso do operador Booleano AND, para localizar a correspondência.

Entretanto, se você clicar em **Corresponder Qualquer (Match Any)**, apenas um dos Critérios de Link precisará ser correspondido, o que equivale a uma operação OR.

Observe que o nome do Atributo a ser correspondido pode ser especificado como uma Expressão (consulte “Expressões” na página 39 para obter detalhes). Os formatos possíveis para o campo Valor de um Critério de Link simples são:

Uma Cadeia de Texto
Mapeada para uma constante com aquele valor.

$Name
Corresponde a `work.getString("Name")`, que é o primeiro valor do atributo *Nome*.

@Name
Corresponde a um dos valores do atributo de multivalores *Nome*.

Uma Expressão IBM Security Directory Integrator
Descrito em detalhes aqui: “Expressões” na página 39.

Critérios de Link Avançados: Você também pode criar os seus próprios critérios de procura customizados, marcando a caixa de opções **Construir critérios com script customizado**. Isso apresenta a você um editor de script para gravar a expressão dos seus próprios Critérios de Link. Nem todos os Conectores suportam os Critérios Avançados de Link e a documentação do Conector informa se eles são suportados. Consulte "Conectores", na publicação *Referência*.

A expressão de procura que você constrói deve estar em conformidade com a sintaxe esperada pelo sistema subjacente. Para transferir sua expressão de pesquisa para o Conector, você deve ocupar o objeto *ret.filter* com sua expressão de cadeia.

Um exemplo simples do JavaScript para um Conector SQL é:

```
ret.filter = " ID LIKE '" + work.getString("Name") + "'";
```

Esses Critérios de Link supõem um exemplo em que a origem de dados possui um atributo chamado *ID* (geralmente um nome de coluna) que desejamos corresponder ao atributo *Name* na Entrada de trabalho.

Nota:

1. A primeira parte da expressão SQL, `Select * from Table Where`, é fornecida pelo IBM Security Directory Integrator.
2. Aspas simples foram incluídas porque `work.getString()` retorna uma cadeia, enquanto a sintaxe SQL pede aspas simples em torno de constantes das cadeias.
3. A sintaxe especial com **$** e **@** não é usada aqui.

Erros de Critérios de Link

O erro mais comum que você obtém quando usa os Critérios de Link é:

```
ERROR> AssemblyLine x failed because
No criteria can be built from input (no link criteria specified)
```

Esse erro ocorre quando você tem um Critério de Link que se refere a um atributo que não pode ser localizado durante a Consulta. Por exemplo, com o seguinte Critério de Vinculação:

```
uid equals $w_uid
```

A configuração de Critérios de Link falhará se *w_uid* não estiver presente na Entrada de trabalho. Isso pode ser devido a eles não serem lidos nas origens de entrada (por exemplo, não em um Mapa de Entrada, ou à ausência da origem de entrada) ou terem sido removidos da Entrada de trabalho em um script. Ou seja, a chamada de função `work.getAttribute("w_uid")` retorna NULL.

Uma maneira de evitar isso é gravar um código em Antes de Executar o Gancho do Conector no modo Consultar, Excluir ou Atualizar que ignora sua operação quando os Critérios de Link não podem ser resolvidos devido à ausência de atributos. Por exemplo:

```
if (work.getAttribute("w_uid") == null)
  system.ignoreEntry();
```

Suas regras de negócios podem exigir outro processamento, tal como uma chamada `skipEntry()` em vez de `ignoreEntry()`, o que faz com que o AssemblyLine pare de processar a entrada atual e comece a partir do início em uma nova iteração. A função ignoreEntry() simplesmente ignora o Conector atual e continua com o restante do AssemblyLine.

Funções

Uma *Função*, geralmente chamada de Function Component (FC), é um componente muito semelhante a um Conector, exceto que não possui uma configuração de modo. Enquanto os Conectores fornecem verbos de acesso padrão para sistemas conectados (Consultar, Excluir, Atualizar e etc), as Funções, por outro lado, desempenham apenas uma operação única, como enviar dados por meio de um Analisador, efetuando dispatch do trabalho para outro AssemblyLine ou fazendo uma chamada de serviço da Web.

Visão Geral

As Funções podem aparecer em qualquer lugar na seção Fluxo de um AL. A pasta da biblioteca Funções no navegador de Configuração pode ser usada para gerenciar sua biblioteca de Componentes de Função.

Tais como os Conectores, as Funções nos AssemblyLines fornecem uma opção Inicializar para determinar quando esse componente inicia. Por padrão, as Funções inicializam durante a inicialização do AL.

Componentes de Script

O Script Component (SC) é um bloco definido pelo usuário do código JavaScript que você pode soltar em qualquer local na lista Fluxo de Dados do AssemblyLine, juntamente com seus Conectores e Componentes de Função, fazendo com que o código de script seja executado para cada ciclo neste ponto do fluxo de trabalho do AL.

Visão Geral

Ao contrário de Ganchos, Componentes de Script são facilmente movidos no fluxo do AssemblyLine, o que os torna ferramentas muito eficientes para qualquer operação, desde a depuração até a criação de protótipos e a implementação de sua própria lógica de fluxo. Além disso, diferente dos outros tipos de componentes do AL, o Script não possui nenhum comportamento ou modo predefinido; você pode implementar o comportamento e o modo com seu script. Uma biblioteca de Scripts pode ser armazenada sob a pasta da biblioteca Scripts no navegador de Configuração.

Uso

Por exemplo, se você quiser testar e depurar apenas parte de um AssemblyLine, poderá colocar o código a seguir em um SC para limitar e controlar o fluxo do AL.

```
task.dumpEntry( work );
system.skipEntry();
```

Quando colocado no ponto apropriado no fluxo de um AssemblyLine, esse SC exibe o conteúdo do objeto de trabalho, gravando-o na saída do log e, então, ignorando o restante do AL nesse ciclo. Movendo o SC para cima e para baixo na lista de componentes, é possível controlar a parte do AL que é realmente executada. Se você trocar a chamada `system.skipEntry()` pela chamada `system.skipTo("ALComponentName" )`, transmitirá o controle diretamente para um Componente do AL específico.

Também é possível usar SCs para conduzir outros componentes. Um cenário comum ao fazer a sincronização de diretórios ou de bancos de dados é a necessidade de manipular informações atualizadas e informações excluídas. Como

os Conectores acionados pelo fluxo de trabalho integrado do AL podem apenas operar em um modo por vez (Atualização ou Exclusão), você precisará estender um pouco essa lógica com seu próprio código. Um método é incluir dois Conectores, um no modo Atualizar e um no modo Excluir, depois, inserir código no Gancho Antes da Execução em cada Conector para indicar a ele para ignorar as operações de mudança que não devem ser tratadas. Por exemplo, no Gancho Antes da Execução do Conector de Atualização, você gravaria algo assim:

```
// The LDAP change log contains an attribute called "changeType"
if (work.getString("changeType").equals("delete"))
 system.ignoreEntry();
```

Isso fará com que o Conector no modo Atualizar ignore Entradas excluídas. Haverá um código complementar no Gancho Antes da Execução do Conector no modo Excluir, ignorando todas as operações, exceto as exclusões.

Entretanto, se você estiver sincronizando atualizações em vários destinos, isso exigirá dois Conectores para cada origem de dados. Outra abordagem é ter um único Conector no estado Passivo e que possa ser acionado a partir de scripts. Por exemplo, digamos que você possua um Conector AL no estado Passivo denominado *synchIDS*. Você pode, então, incluir um SC com o seguinte código para executá-lo:[5]

```
if (work.getString("changeType").equals("delete"))
 synchIDS.deleteEntry( work )
else
   synchIDS.update( work );
```

Contanto que você rotule seu SC claramente, indicando que ele está executando Conectores Passivos, essa abordagem resultará em AssemblyLines mais curtos que serão mais fáceis de serem lidos e mantidos. Este é um exemplo de ter de optar entre duas boas práticas: manter o AL curto e usar uma lógica incorporada versus um script customizado. Entretanto, nesse caso, as metas de clareza e simplicidade são atendidas da melhor maneira com a gravação de um script pequeno.

O Componente de Script também é muito prático quando você deseja testar a lógica e o código de script. Simplesmente crie um AssemblyLine com um único Componente de Script na lista Fluxo de Dados, insira seu código e execute-o.

Consulte Também

Capítulo 2, “Script no IBM Security Directory Integrator”, na página 45.

AttributeMaps

Os Mapas de Atributos são caminhos para os dados fluírem para dentro ou para fora do AssemblyLine. Os Mapas de Atributos aparecem em Conectores e Funções como Mapas de Entrada e de Saída e também estão disponíveis como componentes independentes no AssemblyLine.

Visão Geral

O diagrama no início da seção intitulada “O AssemblyLine” na página 1 representa três Mapas de Atributos como setas curvadas: dois Mapas de Entrada trazendo dados para o AssemblyLine e também um Mapa de Saída passando dados para o cache do Conector (a Entrada *Conn*) para que possam ser gravados.

5. Como os Conectores Passivos não são executados pela lógica do AL, não importa onde eles aparecem na lista Fluxo de Dados.

Cada Mapa de Atributos contém uma lista de regras que criam Atributos na Entrada de Trabalho ou na Entrada Conn. Uma regra de mapeamento especifica duas coisas:

1. O nome do Atributo a ser criado (ou sobrescrito) na Entrada de destino. Esta é a Entrada de Trabalho no caso de Mapas de Entrada e de componentes do Mapa de Atributos independentes, enquanto os Mapas de Saída possuem como destino a Entrada Conn.
2. A designação usada para preencher o Atributo com um ou mais valores. A designação pode ser um script de designação, por exemplo:

   ```
   work.Title
   ```

   ou pode ser um texto literal (incluindo caracteres de nova linha) com substituição de token opcional:

   ```
   <html>
      <header>
         <title>{work.Title}</title>
      </header>
      <body>
         <p>Procure o valor de Atributo "Title" no título desta página</p>
      </body>
   </html>
   ```

Os Mapas de Atributos são criados usando o Config Editor; consulte “Mapeamento de Atributos e Esquema” na página 109.

Os Mapas de Atributos suportam herança, no nível de Mapa e para regras de mapeamento individuais. Observe que você pode arrastar um Script para um Mapa de Atributo para configurar uma regra de mapeamento de JavaScript herdada.

Os Mapas de Atributos também fornecem uma funcionalidade chamada “Comportamento Nulo”, que é usada para controlar como os dados ausentes são tratados.

Consulte “Modelo de Dados Interno: Entradas, Atributos e Valores” na página 46 para obter algumas informações adicionais sobre Mapas de Atributos.

Comportamento Nulo

Ocasionalmente, o sistema tenta mapear um atributo que está faltando. Por exemplo, se um número de telefone opcional não estiver presente na origem de dados de entrada ou um atributo em um Mapa de Saída for removido da Entrada de trabalho. Outras vezes, embora um atributo esteja presente, ele não possui valores - como uma coluna anulável em uma tabela de banco de dados.

Origens de dados diferentes tratam valores ausentes de maneiras diferentes (valor nulo, cadeia vazia) e o recurso descrito nesta seção fornece uma maneira de customizar como atributos ausentes - ou valores de atributos - são tratados. Esse recurso é chamado *Comportamento Nulo* e com ele você pode definir o que é um "valor nulo" e como ele pode ser manipulado .

Nota: O Conector JDBC possui a configuração do parâmetro `jdbcExposeNullValues`, permitindo que você mapeie valores nulos para Atributos ausentes (consulte "Conector JDBC" em *Referência*).

O comportamento nulo pode ser especificado em vários níveis: sistema, Configuração, AssemblyLine, AttributeMap e Atributo. No entanto, como

Comportamento Nulo é altamente específico para origem de dados, faz mais sentido definir essa propriedade de sistema no nível de Mapa de Atributos (por exemplo, para todos os Atributos manipulados por Mapas de Entrada ou Saída do Conector). Esses níveis possíveis são descritos aqui com mais detalhes:

Nível Sistema
A especificação de Comportamento Nulo no nível do sistema é feito no arquivo global.properties configurando as propriedades `rsadmin.attribute.nullBehavior` e `rsadmin.attribute.nullDefinition`, um dos valores listados posteriormente nesta seção.

Nível Config
Isso substitui o Comportamento Nulo no Nível do sistema, e é configurado pela definição das propriedades `rsadmin.attribute.nullBehavior` ou `rsadmin.attribute.nullDefinition` em um Armazenamento de Propriedades para um dos valores a seguir listados posteriormente nesta seção.

Nível AssemblyLine
O Comportamento Nulo do AssemblyLine é especificado clicando no botão **Opções...** da barra de ferramentas do AssemblyLine, selecionando **Configurações do AssemblyLine** e clicando em **Comportamento de Valor Nulo** na caixa de diálogo que é exibida.

Nível do Mapa de Atributos
Definir Comportamento Nulo para todos os Atributos em um mapa é feito clicando no botão **Mais...** acima da lista de mapas de atributos e selecionando **Comportamento Nulo**.

Nível Atributo
Comportamento Nulo pode ser configurado para um atributo específico clicando com o botão direito do mouse nele em um mapa de atributos e selecionando **Comportamento Nulo**. Quando isso for definido para um Atributo, ele será então indicado pela presença de um símbolo de marcador azul no item mapeado.

O Comportamento Nulo suporta cinco configurações diferentes para definir o que é um valor nulo, conforme mostrado abaixo (Observe que o texto entre parênteses de cada configuração é o valor usado para configurar a definição de Comportamento Nulo no nível do sistema, e é geralmente definido no Armazenamento de Soluções ou de Propriedade Global). Cada configuração mostra o valor da propriedade real entre parênteses. Observe que estas definições são listadas na ordem inclusiva, para que o segundo caso também inclua o primeiro; o terceiro inclua os dois primeiros e assim por diante:

Atributo ausente (`AbsentAttribute`)
O Atributo mencionado como a origem de valor(es) em um mapa de atributos está ausente.

Atributo sem valores (`EmptyAttribute`)
O Atributo usado como a origem de valor(es) em um Atributo foi encontrado, mas não tem valores. O caso anterior também verificado.

Atributo contém um valor de cadeia vazia (`EmptyString`)
O Atributo é localizado, mas possui apenas um único valor de cadeia.

Valor (`value`)
O Atributo contém um valor especificado. No AssemblyLine, mapa de atributos e definição de valor nulo no nível do Atributo, esse valor é configurado no campo **Valor** do diálogo Comportamento Nulo. Aqui, é

possível especificar vários valores de atributo se desejado colocando valores em linhas separadas. Se você usar `rsadmin.attribute.nullDefinition` para a configuração nos níveis Sistema e Configuração, também deverá configurar a propriedade `rsadmin.attribute.nullDefinitionValue`.

Nota: Vários aprimoramentos foram feitos no conector do servidor HTTP. Os componentes baseados em TCP, como o Conector do servidor HTTP, possuem uma opção em suas telas de Configuração para retornar cabeçalhos TCP como valores de Atributo. Quando esse sinalizador for limpo, os cabeçalhos TCP são armazenados como propriedades no objeto de entrada retornado.

Comportamento Padrão (`Default Behavior`)
A definição do valor nulo deve ser herdada de um nível superior. Por exemplo, um Atributo herda sua definição de valor nulo da configuração do mapa de atributos, que, por sua vez, a herda do AssemblyLine.

Nota: O Comportamento Nulo em nível Config substitui quaisquer definições em nível Sistema. Além disso, a configuração de Comportamento Padrão no nível do sistema é o mesmo que especificar **excluir**, enquanto no nível de Configuração é equivalente a **valor**.

O recurso Comportamento Nulo permite também definir a ação a ser executada no caso de um valor nulo ser detectado:

Cadeia Vazia (`empty string`)
Os atributos que estão faltando são mapeados com um único valor que possui um valor de Cadeia vazia ("").

Nulo (`null`)
Os atributos que estão faltando são mapeados sem valores, significando que a chamada `att.getValue()` retorna **null**.

Excluir (`delete`)
O atributo é removido do mapa.

Valor (`value`)
Os atributos que estão faltando são mapeados com um valor específico. No AssemblyLine, mapa de atributos e Comportamento Nulo no nível do Atributo, os valores são configurados na edição de **Valor** do diálogo Comportamento Nulo. Aqui, é possível especificar vários valores de atributo se desejado colocando valores em linhas separadas. Se você usar `rsadmin.attribute.nullBehavior` para as configurações no nível Config ou Sistema, também deverá configurar a propriedade `rsadmin.attribute.nullBehaviorValue`.

Comportamento Padrão (`Default Behavior`)
Comportamento Nulo deve ser herdado de um nível superior. Por exemplo, o nível Atributo herda do AttributeMap, que, por sua vez, herda da configuração AssemblyLine.

Nota: O Comportamento Nulo em nível Config substitui quaisquer definições em nível Sistema. Além disso, a definição do Comportamento Padrão em nível Sistema é a mesma especificada em **delete**, enquanto em nível config é equivalente a **value**.

Componentes de Ramificação

Os componentes de ramificação afetam a ordem em que os outros componentes, tais como, Conectores, Scripts, Funções, AttributeMaps e outros Componentes de ramificação, são executados no Fluxo do AssemblyLine.

Visão Geral

Os componentes de ramificação são fornecidos em três variedades:

- Simples (também chamados apenas de Ramificação)
- Loops (Ramificações que efetuam loop)
- Comutadores (Ramificações que compartilham a mesma expressão)

As ramificações podem aparecer em qualquer lugar na seção Fluxo, mas não há nenhuma pasta de biblioteca para elas na seção Recursos de sua área de trabalho.

Uma Ramificação não precisa ser executada até a conclusão; a mesma chamada de script é usada para sair programaticamente de qualquer tipo de Ramificação: `system.exitBranch()`. Consulte “Encerrando uma Ramificação (ou um Loop ou o Fluxo do AL)” na página 30 para obter informações adicionais.

Os três tipos de componentes de Ramificação são:

Ramificação

Embora cada tipo de Ramificação permita definir rotas alternativas para processamento do AssemblyLine, esta forma mais simples determina a ação no caso: "Se esta situação ocorrer, execute esta ação". Você define o que "situação ocorrer" significa configurando as Condições que devem ser atendidas; por exemplo, comparando valores de dados ou verificando o resultado de alguma operação. Se as condições forem verdadeiras, os componentes anexados sob essa ramificação serão executados.

A Ramificação permite definir as Condições com base em quaisquer dados no servidor IBM Security Directory Integrator: Valores de atributos, configurações de parâmetros, propriedades externamente acessíveis e quaisquer informações disponíveis usando JavaScript, como chamadas do sistema operacional para uso de disco ou memória. Várias Condições são vinculadas com a lógica AND ou OR, dependendo da configuração na caixa de opções **Corresponder Qualquer**.

Essa forma mais simples do componente de Ramificação também suporta três configurações de subtipo, IF, ELSE-IF e ELSE, que você pode selecionar.

IF Pode aparecer em qualquer lugar no Fluxo do AssemblyLine. A Ramificação IF fornece uma faixa alternativa para ser seguida pelo processo se as Condições forem verdadeiras. Assim que os componentes sob a Ramificação são executados, o controle passa para o primeiro componente *após* essa Ramificação. Se você não desejar que isso aconteça, deverá incluir uma Ramificação ELSE ou ELSE IF, ou sair da Ramificação com uma chamada de script para `system.exitBranch()`.

ELSE-IF
Idêntica à Ramificação IF, exceto que pode aparecer somente logo após uma Ramificação IF ou ELSE-IF.

ELSE Pode aparecer somente imediatamente após uma Ramificação IF ou

ELSE-IF. A Ramificação ELSE não possui Condições. Seus componentes serão processados apenas se nenhuma Ramificação IF ou ELSE-IF precedente for verdadeira. Além disso, a instância de ELSE é sempre avaliada como true, portanto, não há condições avaliadas durante o ciclo.

Conforme mencionado acima, você pode sair prematuramente de uma Ramificação por meio do script, usando `system.exitBranch()`.

Loop

O componente de Loop fornece a funcionalidade para incluir uma lógica cíclica dentro de um AssemblyLine. Os loops podem ser configurados para três modos de operação: baseado em Condições, baseado em um Conector ou baseado nos valores de um Atributo:

Condicional
Exatamente como uma Ramificação simples, você pode definir Condições que controlam o comportamento do Loop. O Loop continuará a repetir o ciclo enquanto as Condições forem atendidas e parará assim que elas falharem. A janela de detalhes para Loops é a mesma que para as Ramificações simples descritas na seção anterior.

Conector
Esse método permite configurar um Conector no modo Agente Iterativo ou Consulta e fará o ciclo no fluxo do Loop para cada entrada retornada. A janela de detalhes desse tipo de Loop contém as guias do Conector necessárias para configurá-lo, conectar e descobrir atributos e configurar o Mapa de Entrada.

Observe que existe um parâmetro denominado **Opções de Inicialização (Init Options)**, com o qual é possível fornecer instruções ao AL para:

- **Fazer Nada** significa que o Conector não será preparado de nenhuma maneira entre os ciclos do AL.
- **Inicializar e Selecionar/Consultar** faz com que o Conector seja reinicializado para cada ciclo do AL.
- **Apenas Selecionar/Consultar**, que mantém o Conector inicializado, mas refaz a seleção de Repetidor ou a Consulta, dependendo da configuração do Modo.

Observe também que existe uma guia **Parâmetros do Conector (Connector Parameters)**, que funciona de maneira semelhante a um Mapa de Saída, no sentido de que é possível selecionar quais parâmetros de Conector devem ser configurados a partir de valores de Atributos de **trabalho**.

Isso nos leva ao tópico sobre como o Loop com um Repetidor difere disso com base no modo Consultar. Ambas as opções desempenham *procuras* que criam um conjunto de resultados retornado para loop. Para o modo Repetidor, o conjunto de resultados é controlado exclusivamente pelas configurações de parâmetros desse componente. Por outro lado, o modo de consulta usa Critérios de Link para definir regras de procura ou correspondência. Como ele libera você da codificação de Ganchos, como Em Nenhuma Correspondência ou Em Múltiplos

Localizados, este é o modo preferencial para executar procuras que nem sempre podem retornar uma (e apenas uma) entrada correspondida.

Valor de Atributo
Por meio da seleção de qualquer Atributo disponível na Entrada de trabalho, o fluxo de Loop será executado para cada um de seus valores. Cada valor é transmitido ao Loop em um novo atributo Entrada de Trabalho (Work Entry), nomeado no segundo parâmetro. Essa opção permite trabalhar facilmente com atributos com diversos valores, como e-mail ou listas de associações de grupo.

Você pode sair prematuramente de um Loop por meio do script, usando `system.exitBranch()`.

Comutador
Diferente das expressões usadas nas Condições de Ramificações e Loops, a expressão do Comutador pode resultar em mais valores do que apenas `true` ou `false`. Por exemplo, você poderia ativar o valor de um Atributo ou a operação solicitada quando esse AL fosse chamado de um outro AssemblyLine ou processo. Sob o componente Comutador (Switch), você inclui um Caso para cada valor de constante da expressão do Comutador a ser manipulado. Portanto, por exemplo, se você configurasse o Comutador para usar o código de operação delta na Entrada de Trabalho, seus Casos seriam para valores como "add", "delete" e "modify".

Em uma construção de Caso de Comutador do AL, múltiplos casos podem estar ativos ao mesmo tempo. O IBM Security Directory Integrator verifica cada caso, exatamente como faria em uma série de Ramificações IF padrão. O seguinte exemplo mostra como os múltiplos casos funcionam:

```
work.setAttribute("test","abc");

Switch work.test
  Case startsWith("a"): this is true
  Case contains ("bc"): this is true
  Case length=3: this is true
```

As três expressões de Switch work.test que são true acionarão a execução do Comutador.

Você pode sair prematuramente de um Caso de Comutador por meio do script, usando `system.exitBranch();`.

Encerrando uma Ramificação (ou um Loop ou o Fluxo do AL)

Se você desejar sair de uma Ramificação, Loop ou Chave, ou mesmo de Ramificações internas como a seção Fluxo do AL, use o método `system.exitBranch()` a partir de um local em que você possa criar um script, por exemplo, um Gancho ou até mesmo um Componente de Script. Chamar `system.exitBranch()` sem parâmetros (ou com uma cadeia vazia) fará com que a Ramificação contida seja encerrada, e o fluxo continuará com o primeiro componente após essa Ramificação.

Também é possível fornecer ao método um parâmetro de cadeia contendo:

Uma das palavras-chave reservadas: Ramificação, Loop, Fluxo, Ciclo ou AssemblyLine (sem distinção de maiúsculas e minúsculas)
Isso interromperá a primeira Ramificação desse tipo, fazendo um rastreio regressivo até o AssemblyLine. Portanto, se o seu código de script estiver em uma Ramificação dentro de um Loop e você executar a chamada

`system.exitBranch("Loop")`, encerrará tanto a Ramificação quanto o Loop no qual ela está contida. Usar a palavra reservada `Fluxo` faz com que o fluxo saia da seção Fluxo do AssemblyLine, continuando no comportamento de resposta no caso de um Conector de Modo do Servidor ou em um Repetidor ativo a ser lido na próxima entrada ou no encerramento do AL (Epílogos, ...). A palavra-chave `Ciclo` passa o controle ao final do ciclo do AL atual e não chama o comportamento de resposta em Conectores no Modo Servidor, enquanto a palavra-chave `AssemblyLine` fará com que o AL seja interrompido e encerrado.

Todos os outros valores da chamada `system.exitBranch()` causam uma quebra da ramificação/loop que tem o nome especificado. Portanto, por exemplo, a chamada `system.exitBranch("IF_LookupOk")` envia o fluxo após a Ramificação ou o Loop contido chamado "IF_LookupOk". Observe que diferentemente de `system.skipTo()`, que passará o controle a qualquer componente do AL nomeado, `system.exitBranch()` fará com que o processamento continue após o Loop/Ramificação especificado.

O nome de uma Ramificação ou Loop (com distinção entre maiúsculas e minúsculas)

Se você transmitir o nome de uma Ramificação ou de um Loop em que a chamada de script está aninhada, o controle será transmitido ao componente seguinte no AL. Se nenhuma Ramificação ou Loop com esse nome for encontrado, o rastreio será invertido a partir do ponto da chamada, e então resultará em um erro.

Há também uma funcionalidade *continue* nos Componentes Loop. Os seguintes métodos estão disponíveis no objeto do sistema:

```
system.continueLoop();
system.continueLoop(name);
```

em que *name* é uma sequência sem distinção a maiúsculas e minúsculas, indicando um nome de Loop. No caso de um nome de Loop ser fornecido, o fluxo do programa é transferido para o Componente de Loop com esse nome.

Analisadores

Os analisadores são usados em conjunto com um componente de fluxo de bytes, por exemplo, um Conector do Sistema de Arquivos, para interpretar ou gerar a estrutura do conteúdo que está sendo lido ou gravado.

Observe que quando o fluxo de bytes que você está tentando analisar não estiver em harmonia com o Analisador escolhido, uma `sun.io.MalformedInputException` será recebida. Por exemplo, essa mensagem de erro pode ser mostrada ao usar a guia **Mapa de Entrada** para pesquisar um arquivo.

O Config Editor fornece dois locais onde você pode selecionar Analisadores:

1. Na guia **Analisador** de um Conector de fluxo de bytes.
2. A partir de seus próprios scripts; por exemplo, Ganchos e componentes de script.

Para obter informações adicionais sobre Analisadores individuais, consulte "Analisadores", na publicação *Referência*.

Conversão de Codificação de Caracteres

O Java2 usa Unicode como sua codificação de caracteres internos. Unicode é um conjunto de caracteres de duplo byte. Quando você trabalha com cadeias e caracteres em AssemblyLines e Conectores, eles são sempre assumidos para estar

em Unicode. A maioria dos Conectores fornece alguns meios de conversão de codificação de caracteres. Quando você lê a partir dos arquivos de texto no sistema local, Java2 já estabeleceu uma conversão de codificação de caracteres padrão que depende da plataforma na qual você está executando.

O servidor IBM Security Directory Integrator possui a opção da linha de comandos **-n**, que especifica o conjunto de caracteres de arquivos de configuração que ele usará ao gravar novos; integra também esse designador do conjunto de caracteres ao arquivo, de modo que possa interpretar corretamente o arquivo ao lê-lo novamente mais tarde.

No entanto, ocasionalmente, você lê ou grava dados de ou para arquivos de texto nos quais as informações são codificadas em diferentes codificações de caracteres. Por exemplo, os Conectores que exigem um Analisador geralmente aceitam um parâmetro **Conjunto de Caracteres** na configuração do Analisador. Esse parâmetro deve ser definido como uma das tabelas de conversão aceitas, conforme especificado pelo IANA Charset Registry (http://www.iana.org/assignments/character-sets).

Alguns arquivos, quando codificados em UTF-8, UTF-16 ou UTF-32, podem conter um BOM (Byte Order Marker) no início do arquivo. Um BOM é a codificação dos caracteres `0xFEFF`. Ele pode ser utilizado como assinatura para a codificação utilizada. Entretanto, o Conector de Arquivos do IBM Security Directory Integrator não reconhece um BOM.

Se você tentar ler um arquivo com um BOM, deverá incluir esse código, por exemplo, Antes do Gancho de Seleção do conector:

```
var bom = thisConnector.connector.getParser().getReader().read(); // skip the BOM = 65279
```

Esse código fará a leitura e ignorará o BOM, supondo que você especificou o conjunto de caracteres correto para o analisador.

Deve-se tomar cuidado com o protocolo HTTP; consulte o *Referência*, na seção sobre codificação de conjuntos de caracteres da descrição do Analisador HTTP para obter mais detalhes.

Acessando suas Próprias Classes Java

Você pode acessar suas próprias classes Java customizadas de dentro da estrutura do IBM Security Directory Integrator enquanto elas forem classes e métodos *públicos*. Essas bibliotecas devem ser compactadas em um arquivo .jar ou .zip e, em seguida, colocadas no diretório `TDI_install/jars`, de preferência em seu próprio subdiretório. Você pode também usar a variável de ambiente CLASSPATH ou a pasta de extensão de ambiente de tempo de execução Java, mas esses dois métodos são desaconselháveis. Esses métodos permitem chamar classes de dentro de suas próprias classes apenas se acontecer de o utilitário de carga carregar as classes antes de você próprio.

Se você estiver executando o servidor a partir do Editor de Configuração, deverá reiniciar o Editor de Configuração antes de ele detectar novas classes no diretório e nos subdiretórios `TDI_install/jars`.

Depois de colocar os arquivos .jar no subdiretório jars, você poderá criar uma instância da classe para referência dentro do IBM Security Directory Integrator. Observe que o Componente da Função Java permite abrir arquivos .jar, navegar por objetos contidos neles, bem como por seus métodos. Depois que você escolhe a

função a ser chamada, o FC prepara seu esquema de Entrada e Saída para corresponder os parâmetros necessários à função Java.

Para obter informações adicionais sobre chamada de classes Java a partir do script, consulte “Instanciando uma Classe Java” na página 75.

Instanciando as Classes com o Uso do Config Editor

Utilize a pasta Bibliotecas Java da janela **Configurações e Criação de Log de Solução** no Editor de Configuração para declarar suas classes. Isso funcionará apenas se sua classe tiver um contrutor sem argumento, que é geralmente, mas nem sempre, o construtor padrão.

Ao incluir um objeto de classe, clique em **Incluir...** e especifique dois parâmetros: o nome do objeto do script, ou seja, o nome da variável de script que é uma instância de sua classe Java e o nome da classe Java. Por exemplo, é possível ter um Nome de Objeto de Script *mycls* enquanto a Classe Java pode ser *my.java.classname*. O objeto *mycls* estará disponível para quaisquer AssemblyLines definidos antes da execução dos Prólogos Globais.

Nota: Observe que isso faz com que o objeto seja instanciado para cada execução do AssemblyLine. Se isso não for o desejado e se você preferir instanciar em demanda, consulte a próxima seção.

Instanciar as Classes em Tempo de Execução

Se você desejar instanciar sua classe em um ponto específico de execução ou para as classes que não possuem construtores sem argumentos, será necessário instanciar a classe durante o tempo de execução. Por exemplo:

```
crytoLib = new com.acme.myCryptoLib();
```

Fluxo do AssemblyLine e Ganchos

AssemblyLines fornecem um comportamento automatizado incorporado que ajuda a criar e a implementar fluxos de dados rapidamente. Esse comportamento automatizado é detalhado nos Diagramas de Fluxos do *Referência*. Além disso, Conectores e Funções têm seus próprios comportamentos e eles são mostrados nos Diagramas de Fluxos. Observe que o comportamento do Conector depende da configuração do Modo.

Em todos esses fluxos lógicos internos há vários *waypoints* em que você pode incluir sua própria lógica com script para estender o comportamento interno ou substituí-la completamente. Esses waypoints são chamados de "Ganchos" e estão disponíveis para customização na guia Ganchos de todos os Conectores e Funções, bem como a partir do próprio AssemblyLine.

Os Ganchos poderão ser ativados e desativados dependendo se um Gancho específico for aplicável ao AssemblyLine que estiver sendo executado. Ao desativar um Gancho, você não interrompe sua herança no conector do qual ele faz parte.

Quando um AssemblyLine é ativado, ele passa por três fases: Inicialização, Fluxo de Dados e Encerramento. Durante a Inicialização, Ganchos de Prólogo estão disponíveis para reconfigurar os componentes antes deles serem inicializados. Na fase Fluxo de Dados, cada Entrada de Trabalho alimentada no AssemblyLine passa pelos componentes de Fluxo para processamento.

Por último, durante o Encerramento, Ganchos de Epílogo podem ser usados para executar trabalho de fim de tarefa, como verificar e relatar status do erro ou armazenar dados de estado para a próxima vez em que o AL for iniciado.

Fase de Inicialização

Nesse ponto, o servidor é instruído a carregar e executar um AssemblyLine. O servidor usa o projeto armazenado no arquivo de configuração para configurar o AL. Se um TCB (TaskCallBlock) tiver sido transmitido para o AssemblyLine, seu conteúdo será avaliado (o que pode resultar em alterações nos parâmetros do componente do AL). Neste ponto, os fluxos do Gancho de Prólogo são iniciados.

Prólogos Globais

Primeiro, se algum Prólogo Global estiver definido, então eles serão avaliados. Os Prólogos Globais são Scripts na pasta Recursos no Projeto que foram incluídos (na janela **Configurações e Criação de Log de Solução**) no AssemblyLine. Isso é geralmente feito na janela Configurações do AssemblyLine selecionando os Scripts para serem executados na inicialização do AL. Após a conclusão dos Prólogos Globais, os Ganchos de Prólogo do AL são chamados.

Ganchos de Prólogo do AL (Antes de Inicializar)

Primeiro, o Prólogo do AssemblyLine - Antes de Inicializar o Gancho é chamado. Depois disso, todos os Conectores e Funções configurados para Inicializar "na Inicialização" passam para a fase de inicialização, que chama também seus Ganchos de Prólogo, como visto no próximo ponto.

Inicialização de Conector/Função

A sequência de inicialização é executada para cada Conector e Função com Inicialização definida como "na Inicialização". Eles são iniciados um de cada vez, conforme definido pela ordem no AssemblyLine. Para cada Conector ou Função, o fluxo é o seguinte:

1. O Gancho **Prólogo – Antes de Inicializar** do componente é chamado.
2. O componente é iniciado; por exemplo, conectando à sua origem de dados, destino ou API subjacente.
3. Para Conectores no modo Repetidor, o Gancho **Prólogo – Antes da Seleção** é processado e o Conector executa a seleção de entrada; isso ativa uma chamada específica da origem de dados, como executar um SQL SELECT ou uma pesquisa LDAP.
4. Para Repetidores, o Gancho de **Prólogo – Após Seleção** é avaliado.
5. O Gancho de **Prólogo – Depois de Inicializar** é chamado, encerrando a sequência.

Se a inicialização de um Conector falhar, o fluxo do AssemblyLine será transmitido ao Gancho de **Prólogo – No Erro**, onde esse erro poderá ser tratado.

O recurso Reconectar permite configurar um Conector para tentar restabelecer automaticamente sua conexão se ocorrer um erro durante o acesso a dados ou configuração. Essas configurações estão localizadas sob a guia **Falha de Conexão** do Conector.

Nota: Conectores de Script, ou seja, Conectores implementados usando JavaScript, são avaliados nesta etapa, de modo que as funções do Conector necessárias são registradas e o código de inicialização é executado.

Ganchos de Prólogo do AL (Depois de Inicializar)
O Gancho de **Prólogo do AssemblyLine - Depois de Inicializar** é executado. A conclusão desse Gancho indica o fim da Fase de Inicialização e o início da Fase de Fluxo de Dados.

Fase de Fluxo de Dados

Gancho do AssemblyLine Início de Ciclo
Esse Gancho é chamado no início de cada ciclo antes dos componentes Alimentações (Feeds) ou Fluxo (Flow).

Ciclo do AssemblyLine
O controle é passado para o primeiro componente no fluxo, geralmente um Conector no modo Servidor ou Repetidor na seção Alimentações.

Se você tiver um ou mais Repetidores no AssemblyLine, então o primeiro começa o ciclo recuperando a próxima entrada a partir do seu conjunto de resultados e mapeando Atributos na Entrada de Trabalho. A Entrada de Trabalho resultante é transmitida para os componentes da seção Fluxo, iniciando no topo da lista, conforme visto no CE.

Para Conectores no modo Servidor, um processo de listener é ativado, o qual aguarda conexões do cliente de entrada. Quando um pedido de conexão é detectado, o Conector faz um clone de si mesmo, aceita a conexão e, em seguida, comuta a si mesmo para o modo Repetidor, a fim de alimentar dados do cliente na seção Fluxo para processamento. De qualquer maneira, você obtém um Repetidor conduzindo Entradas de Trabalho para os componentes Fluxo. (Enquanto isso, o Conector original no modo Servidor aguarda por pedidos de conexão de entrada.)

Se o AssemblyLine não tiver um Conector no Modo Repetidor ou no Modo Servidor, você terá um AssemblyLine de disparo único geralmente usado para processar uma alimentação de IWE (Entrada de Trabalho Inicial) por um outro processo de chamada.

Fim do Ciclo
Quando o último componente do Fluxo é executado, pode ocorrer um dos seguintes:

- Se a Entrada de Trabalho atual for proveniente de um Repetidor, o controle será transmitido novamente para o Repetidor para obter a próxima entrada de sua origem.
- No caso de um Conector no modo Servidor, uma resposta é enviada ao cliente.
- Para um AL que é chamado no modo de ciclo manual, o encadeamento é transmitido novamente ao responsável pela chamada para que os resultados possam ser acessados.

Não existe um Gancho específico neste ponto, embora ele possa ser incluído em seu AssemblyLine inserindo um Script no final.

Fim dos Dados
Fim dos dados consiste em um Gancho no modo Repetidor que é

chamado quando o fim do conjunto de dados de entrada é alcançado. Neste ponto, o controle é passado para o próximo Conector de Alimentações ou o AssemblyLine entra na Fase de Encerramento.

Fase de Encerramento

Neste ponto, o processamento do AL foi concluído normalmente ou interrompido devido a um erro.

Gancho do AssemblyLine Epílogo - Antes de Fechar

O Gancho do AssemblyLine chamado **Epílogo – Antes de Fechar** é processado.

Fluxo de Fechamento de Conectores/Funções

Os Ganchos de Epílogo de cada Conector e Função são chamados a fim de que apareçam no Editor de Configuração:

1. O Gancho **Antes de Fechar**.
2. A operação de fechamento é executada; por exemplo, fechar uma conexão ou liberar uma chamada de API.
3. O Gancho **Após Fechar**.

Gancho do AssemblyLine Epílogo - Após Fechar

Finalmente, o Gancho do AssemblyLine **Epílogo – Após Fechar** é executado.

Configuração de Conectores no Modo Servidor

Quando um Conector no modo Servidor for iniciado, ele vai para o evento no modo que atende ao modo. Depois que um evento é recebido, ele clona o AL e retoma a espera de eventos adicionais. Enquanto isso, no clone, o Conector comuta a si mesmo para o modo Repetidor e passa o controle ao próximo componente da lista Alimentações. Esse processo permite que você tenha múltiplos Conectores no modo Servidor ativos e alimentação de dados no fluxo ao mesmo tempo — um exemplo seria ter vários Conectores no servidor HTTP no modo Servidor que atende em portas diferentes, mas alimentado no mesmo AL. Embora os Conectores do modo Servidor façam parte de uma configuração do AssemblyLine, eles são executados em processos separados, como encadeamentos.

Existe um conjunto adicional de Ganchos que é avaliado para os Conectores nesse modo. Os Ganchos específicos da funcionalidade do modo Servidor para lidar com conexões de entrada são:

Antes de aceitar a conexão

Esse Gancho é chamado antes de um Conector entrar no modo de atendimento.

Depois de aceitar a conexão

Após o recebimento de uma conexão, esse Gancho é chamado. Observe que não há dados disponíveis nesse momento. Para examinar as informações de eventos de entrada, use os Ganchos do modo Repetidor, como **Depois de GetNext** ou **Êxito em GetNext**.

Erro ao aceitar a conexão

Esse Gancho é executado se ocorrer um erro em qualquer dos Ganchos no modo Servidor ou recebidos da origem de dados durante o atendimento do evento.

- Conforme mencionado anteriormente, se você tiver mais de um Conector no modo Repetidor (consulte "Vários Repetidores em um AssemblyLine" na página 7), esses Conectores são empilhados na ordem em que são exibidos na

configuração, de cima para baixo. Por exemplo, se você tiver dois Repetidores, **a** e **b**, **a** é chamado até que não retorne mais entradas antes que o AssemblyLine alterne para **b**.

- Se você não tiver Conectores no modo Repetidor e nenhuma Entrada de Trabalho Inicial (IWE) for fornecida para o AssemblyLine quando ele for iniciado, por exemplo, por uma chamada de outro AssemblyLine, e se nenhuma Entrada de Trabalho for criada em um Gancho do AssemblyLine ou de Prólogo do Conector, então o AssemblyLine ainda executa uma transmissão simples.

Finalmente, há um Gancho de **Pedido de Encerramento** em que você pode colocar o código que é processado se o AssemblyLine for fechado corretamente devido a um pedido externo para encerrar (ao contrário do travamento), permitindo que você faça com que ele execute um encerramento correto.

Funções especiais estão disponíveis a partir do objeto *system* para ignorar ou tentar novamente a Entrada de trabalho atual, bem como ignorar um Conector, etc. Consulte "Controlando o Fluxo de um AssemblyLine" na página 38 para obter detalhes adicionais.

Manipular Encerramento e Limpeza de Erros Críticos

Há diversos métodos que permitem detectar e manipular erros internos do IBM Security Directory Integrator, bem como erros que ocorrem em Conectores IBM Security Directory Integrator, Analisadores, Componentes de Função etc. Esses métodos incluem:

- Ganchos de Erro, nos quais o código JavaScript pode ser gravado para manipular um erro. Esse método é acessível para usuários do IBM Security Directory Integrator. Além disso, consulte "Controlando o Fluxo de um AssemblyLine" na página 38.
- Blocos Java finalmente tentar capturar, que asseguram que uma falha menor não interrompe o Servidor, bem como que todos os erros são manipulados corretamente. Esses blocos já estão no local nas classes de servidor do IBM Security Directory Integrator principal.

O recurso do Gancho de encerramento JVM aprimora a confiabilidade do Servidor. Os ganchos de encerramento do Java permitem que uma parte do código desempenhe algum processamento após o pressionamento de Control-C ou quando a JVM está sendo encerrada por alguma outra razão, até mesmo System.exit.

Você pode especificar um programa externo para ser iniciado quando o JVM for encerrado. Esse programa externo é iniciado a partir do Gancho de encerramento da JVM. Esse programa externo é configurado usando uma propriedade opcional do arquivo `global.properties` ou `solution.properties`:

```
jvm.shutdown.hook=<executável de aplicativo externo>
```

Os scripts de shell e os arquivos em lote também podem ser especificados como o valor dessa propriedade.

Quando o Gancho de encerramento da JVM é chamado, nada pode ser feito para evitar a finalização da JVM. No entanto, com a execução de um programa externo é possível executar operações customizáveis: por exemplo, enviar uma mensagem que o servidor IBM Security Directory Integrator tenha encerrado, transportando operações de limpeza ou mesmo reiniciando um novo servidor, se desejar.

Controlando o Fluxo de um AssemblyLine

Ganchos são waypoints programáveis no comportamento automatizado interno do IBM Security Directory Integrator, em que você pode impor sua própria lógica.

Os Ganchos são encontrados em AssemblyLines, Conectores e Componentes de Funções. Por exemplo, se desejar ignorar ou reiniciar inteiramente partes do AssemblyLine, você normalmente faz isso a partir de um Gancho em um Conector:

Nota: As construções a seguir podem ser usadas para encerrar um Componente de Ramificação ou um Loop.

system.ignoreEntry()
Ignora o Conector atual e continue processando sua entrada de dados existente com o próximo Conector.

system.skipEntry()
Ignora (elimina) a entrada completamente, interrompendo o ciclo atual, retorna o controle ao início do AssemblyLine e obtém a próxima entrada a partir do Repetidor atual.

system.exitFlow()
Elimina qualquer processamento adicional da entrada atual, executa a lógica de fim de ciclo; por exemplo, salva a Chave de Estado do Repetidor (se o Conector estiver configurado para isso), retorna o controle ao início do AssemblyLine e obtém a próxima entrada do Repetidor atual.

system.restartEntry()
Reinicia do começo do AssemblyLine, forçando o Repetidor atual a reutilizar a entrada atual.

system.skipTo(String name)
Ignora o Conector com nome.

system.abortAssemblyLine(String reason)
Interrompe todo o AssemblyLine com a mensagem de erro específica.

Nota: Se você colocar qualquer código em um **Gancho de Erro** e não terminar o AssemblyLine ou o EventHandler atual, então o processamento continuará, independentemente de como você obteve o Gancho de Erro. Isso significa que mesmo os erros de sintaxe no seu script são ignorados. Então, tenha certeza de verificar o objeto de *error* se desejar saber o que causou o erro.

Os métodos descritos na lista anterior podem estar relacionados como goto-statements, em que nenhum código adicional é executado nesse Gancho. Por exemplo:

```
system.skipEntry(); // Causes the flow to change
// This next line is never executed.
task.logmsg("This will never be reached");
```

Nota: Há uma diferença entre um Gancho de erro inexistente e um vazio - embora nem sempre seja fácil reconhecer no Editor de Configuração. Um Gancho de erro *vazio* faz com que o sistema redefina a condição de erro que fez com que o Gancho seja chamado, após o qual o servidor continua o processamento, enquanto que um Gancho *inexistente* ou indefinido faz com que o sistema execute a manipulação de erros padrão (geralmente interrompendo o AssemblyLine).

Expressões

O IBM Security Directory Integrator fornece o recurso Expressões compatível com a v.6 que permite calcular parâmetros e outras configurações no tempo de execução, tornando suas soluções dinamicamente configuráveis. Esse recurso é expandido na manipulação de Propriedades localizada em versões anteriores.

Além de suportar referências simples de Propriedades Externas (totalmente compatíveis com versões anteriores), o recurso Expressões fornece mais poder ao manipular o AssemblyLine e definições de configuração do componente durante a inicialização e a execução do AL ou do componente. As expressões também podem ser utilizadas para mapas de Atributos, assim como para Condições e Critérios de Link, aliviando bastante os scripts requeridos anteriormente para construir soluções configuradas dinamicamente. O IBM Security Directory Integrator fornece um Editor de Expressão para facilitar a construção dessas expressões.

O recurso Expressões é criado na parte superior dos serviços fornecidos pela classe Java `java.text.MessageFormat` padrão. A classe MessageFormat fornece eficientes recursos de substituição e formatação. Aqui está um link para uma página online que descreve essa classe e seus recursos: http://docs.oracle.com/javase/1.6.0/docs/api/java/text/MessageFormat.html.

Nota: As Expressões baseadas em MessageFormat mostradas nesta seção eram a base de substituição de parâmetro no IBM Security Directory Integrator v.6; na v.7, a melhor prática é usar expressões Avançadas (JavaScript) em vez disso.

Além dos recursos descritos na classe acima, o IBM Security Directory Integrator fornece diversos objetos de tempo de execução que podem ser usados em expressões - apesar de a disponibilidade de alguns objetos depender do estado do tempo de execução; por exemplo, se *conn* ou *current* está definido ou a entrada *error*. A sintaxe de Expressões fornece uma notação de abreviação para acessar as informações nesses objetos, como Atributos em um objeto de entrada nomeado, ou um parâmetro específico de um componente.

Tabela 2. Objetos de Script, seu Uso e Disponibilidade.

Referência do IBM Security Directory Integrator	Valor	Disponibilidade

work.*attrname[.index]*	A entrada *work* no AssemblyLine atual.
	
	O *índice* opcional refere-se ao *enésimo* valor do atributo. Caso contrário o primeiro valor é utilizado.
	
	Este Mapa de atributos avançado:
	`ret.value = work.getString ("givenName") + " " + work.getString("sn");`
	
	pode ser expresso simplesmente como:
	`{work.givenName} {work.sn}`
		AssemblyLine

conn.*attrname[.index]*	A entrada *conn* do AssemblyLine atual. O *índice* opcional refere-se ao *enésimo* valor do atributo. Caso contrário o primeiro valor é utilizado.	AssemblyLine durante o mapeamento de atributo
current.*attrname[.index]*	A entrada *current* no AssemblyLine atual O *índice* opcional refere-se ao *enésimo* valor do atributo. Caso contrário o primeiro valor é utilizado.	AssemblyLine durante o mapeamento de atributo para Modify

Tabela 2. Objetos de Script, seu Uso e Disponibilidade. (continuação)

Referência do IBM Security Directory Integrator	Valor	Disponibilidade
config.*param*	O objeto configuração do AL do componente atual. Além disso, se *config* for usado no parâmetro de um Conector, Analisador ou Função, então ele se refere à *Interface* do objeto Config desse componente, por exemplo, Conector JDBC ou Analisador XML. *param* é o nome do próprio parâmetro, como se você fosse fazer uma chamada para getParam() ou setParam(). Por exemplo, para o Conector JDBC, você poderia fazer a seguinte referência: `{config.jdbcSource]`	AssemblyLine EventHandler Conector Analisador Componente de Função

alcomponent.*name.param*	O valor de parâmetro da Interface do componente de um componente AssemblyLine nomeado.
	
	name é o nome do componente AssemblyLine
	
	param é o nome do parâmetro do objeto *name*
	
	Portanto, a seguinte Expressão:
	`{alcomponent.DB2conn.jdbcSource}`
	
	é equivalente à seguinte chamada em script:
	`DB2conn.connector.getParam ("jdbcSource");`
		AssemblyLine

property[:storename].name property[:storename/bidi].name	Uma referência *TDI-Properties*. O storename opcional aponta um Armazenamento de Propriedades específico. Se nenhum storename for especificado, o armazenamento padrão será utilizado. *name* é o nome da propriedade *bidi* definirá, quando presente, o valor do parâmetro para avançar pela chamada até o Armazenamento de Propriedade referido. Quando *bidi* está presente, nenhum outro padrão de substituição ou texto é permitido.	Sempre
JavaScript<<EOF script code ... // Must contain "return" EOF **Nota:** Sintaze da v.6; no lugar, use a opção Avançado (JavaScipt) no Editor de Expressão	Código de script *integrado* usado para gerar um valor para a Expressão. Esse script deve retornar um valor. O texto "EOF" usado aqui é uma sequência arbitrária que termina o trecho JavaScript. O JavaScript é coletado até que uma única linha com a sequência EOF seja encontrada ou nenhum EOF que esteja sinalizado seja definido - veja a nota abaixo. Observe que JavaScript integrado é avaliado usando a instância do mecanismo de script do AssemblyLine, portanto você tem acesso a todas as variáveis por outro lado presentes no script. **Nota:** Existe um formulário de abreviações de inclusão de JavaScript que funciona para campos de entrada que não suportam múltiplas linhas (como Critérios de Link ou os nomes de Atributos em mapas) e podem portanto não ter a linha EOF necessária: `{JavaScript return work.givenName + " " + work.surName}`	Sempre

JavaScript integrado em Expressões tem acesso ao mecanismo de script do AssemblyLine. Como resultado, mesmo as variáveis de script definidas em outro lugar no AssemblyLine podem ser acessadas. Observe que se você se referir à variável ou ao objeto que não seja um daqueles especificamente listados nas tabelas mostradas nesta seção, o avaliador Expressão verificará com o mecanismo de script do AL para ver se ele está definido lá.

Expressões em Parâmetros de Componente

Quando utilizados para um parâmetro de componente, os seguintes objetos são de interesse especial:

Tabela 3. Objetos Especiais Utilizáveis em Expressões

Objeto	Valor
config	O objeto de configuração da Interface do componente.
mc	O objeto MetamergeConfig da instância Config (`config.getMetamergeConfig()`).
work	A Entrada de trabalho do AssemblyLine.
tarefa	O objeto do AssemblyLine.

Como um exemplo, tome um Conector JDBC com o parâmetro Nome de Tabela definido como "Contas". Você poderia então clicar no rótulo do parâmetro **SQL Select** ou no botão **Abrir diálogo de valor do parâmetro** adjacente ao campo , escolher **Texto c/ substabelecimento** e, em seguida, inseri-lo no campo de texto grande na parte inferior do diálogo:

```
select * from {config.jdbcTable}
```

Isso obterá o parâmetro Table Name e criará a seguinte instrução SQL Select:

```
select * from Accounts
```

Ou poderia avançar mais e tentar algo como isto para o parâmetro SQL Select:

```
SELECT {JavaScript<<EOF

 var str = new Array();
 str[0] = "A";
 str[1] = "B";
 return str.join(",");
EOF

} FROM {property:mystore.tablename} WHERE A = '{work.uniqueID}'
```

O JavaScript integrado retornará o valor “A,B” que é utilizado para concluir o restante da Expressão. Se você tiver um Armazenamento de Propriedade chamado *mystore* com uma propriedade *tablename* definida como “Contas” e um Atributo *uniqueID* na Entrada de trabalho com o valor igual a “42”, o resultado final será:

```
SELECT A,B FROM Accounts WHERE A = '42'
```

Esse resultado avaliado não é exibido no CE. Simplesmente digitar chaves não fará com que a avaliação da Expressão seja feita para o valor de parâmetro. Em vez disso, você terá duas opções ao digitar Expressões para parâmetros:

1. Pressione o botão **Abrir diálogo de valor do parâmetro** adjacente ao campo (ou clique no rótulo Parâmetro) e selecione **Texto c/ substabelecimento** para abrir o diálogo Expressões enquanto estiver no campo de entrada do parâmetro. Você pode entrar sua Expressão no campo de texto grande deste diálogo. Clique em **OK** para inserir sua Expressão.
2. Digite manualmente o preâmbulo especial, @SUBSTITUTE, no campo de entrada de parâmetro, seguido pela Expressão. Por exemplo:

   ```
   @SUBSTITUTEhttp://{property.myProperties:HTTP.Host}/
   ```

   Nota:

   - Esse último método de inserir expressões diretamente não é recomendado; em vez disso, use o Editor de Expressão.
   - Se **Texto com substituição** estiver selecionado como o valor do caminho de arquivo do File Connector e `{work.fullPath}` for inserido, então ocorrerá o erro a seguir: "CTGDIC114E O parâmetro 'Caminho do Arquivo' é necessário." Esse resultado é esperado porque supõe-se que o Editor de Configuração exiba o resultado da aplicação da substituição no texto especificado. Nesse caso, não existe o objeto `work`, pois o objeto `work` é definido apenas no AssemblyLine. Por essa razão, o resultado é uma sequência vazia. Para este parâmetro, uma sequência vazia é um erro e, portanto, a mensagem de erro é mostrada. Entretanto, quando o AssemblyLine estiver em execução, poderá existir um objeto `work` e a avaliação produziria a sequência de parâmetros, conforme necessário.

Expressões em Critérios de Link

As Expressões em Critérios de Link fornecem uma lista semelhante de objetos predefinidos. Novamente, observe que você também possui acesso a quaisquer outros objetos ou variáveis atualmente definidos no mecanismo de script do AssemblyLine.

Tabela 4. Objetos Predefinidos para Uso em Expressões em Critérios de Link

Objeto	Significado
config	O objeto de configuração da Interface do componente
mc	O objeto MetamergeConfig da instância de configuração (`config.getMetamergeConfig()`)
work	A Entrada de trabalho do AssemblyLine
tarefa	O próprio componente ou um componente nomeado
alcomponent	O Componente Conector ou Função

Portanto, por exemplo, suponha que você deseja configurar os Critérios de Link para um Conector de modo que o Atributo a ser usado na correspondência seja determinado no tempo de execução. Além disso, para Atributos de dados padrão da Entrada de trabalho, há também um Atributo *matchAtt* com o valor de sequência "uid". Nesse caso, a seguinte Expressão é usada nos Critérios de Link:

```
{work.matchAtt}  EQUALS {work.uid}
```

é equivalente a isto:

```
uid  EQUALS $uid
```

Expressões em Ramificações, Loops e Comutador/Caso

Esta lista de objetos de Expressão é semelhante àquela de Critérios de Link:

Tabela 5. Objetos Predefinidos para Uso em Expressões em Componentes de Ramificação

Objeto	Significado
config	O objeto de configuração da Interface do componente
mc	O objeto MetamergeConfig da instância Config (`config.getMetamergeConfig()`)
work	A Entrada de Trabalho do AssemblyLine
tarefa	O AssemblyLine
alcomponent	O Componente Conector ou Função

Você pode usar Expressões para o nome Atributo e o Operando de uma Condição. Também pode usar Expressões para configurar os componentes Comutador e Caso.

Criando Script com Expressões

Você pode usar também Expressões diretamente do código JavaScript. Aqui está um exemplo que cria uma Expressão usando a nova classe ParameterSubstitution:

```
var ps = new com.ibm.di.util.ParameterSubstitution("{work.FullName} -> {work.uid}");

map = new java.util.HashMap();

map.put("mc", main.getMetamergeConfig());
map.put("work", work);

task.logmsg(ps.substitute(map));
```

A expressão que resulta do código JavaScript emite as seguintes mensagens de log quando executada para várias iterações no AssemblyLine de teste:

```
14:35:29  Patty S Duggan -> duggan
14:35:29  Nicholas P Butler -> butler
14:35:29  Henri T Deutch -> deutch
14:35:29  Ivan L Rodriguez -> rodriguez
14:35:29  Akhbar S Kahn -> sahmad
14:35:29  Manoj M Gupta -> gupta
```

Objeto de Entrada

Uma das bases da compreensão do IBM Security Directory Integrator é saber como os dados são armazenados e transportados dentro do sistema. Isso é feito usando um objeto chamado *entrada*. O objeto Entrada pode ser considerado um "depósito Java" que pode conter qualquer número de Atributos: nenhum, um ou vários.

Os atributos são também objetos parecidos com depósitos no IBM Security Directory Integrator. Cada Atributo pode conter zero ou mais *valores*; sendo esses os valores de dados reais que são lidos e gravados nos sistemas conectados. Os valores de atributos são também objetos Java; eles podem ser sequências, números inteiros e carimbos de data e hora; seja lá o que for necessário para corresponder ao tipo nativo desse valor de dados. Um único Atributo pode facilmente conter valores de diferentes tipos. Entretanto, os valores de um único Atributo tendem a ser do mesmo tipo na maioria das origens de dados.

Embora esse paradigma *entrada-atributo-valor* corresponda bem ao conceito de entradas de diretório LDAP (Lightweight Directory Access Protocol), é também como as linhas nos bancos de dados são representadas dentro do IBM Security Directory Integrator, como são os registros em arquivos, documentos IBM Lotus Notes e páginas HTTP recebidas pela rede. Todos os dados, de qualquer origem com o qual o IBM Security Directory Integrator trabalhe, são armazenados internamente como objetos de entrada com Atributos e seus valores.

Ao contrário das versões anteriores do IBM Security Directory Integrator, a partir da v7.0, objetos de Entrada hierárquica são suportados, no AssemblyLine e por alguns dos componentes que podem fazer parte de um AssemblyLine. O objeto de Entrada é estendido para fornecer vários métodos convenientes para lidar com dados hierárquicos, embora, por padrão, eles sejam ocultos e sejam reproduzidos somente se você ativá-los explicitamente ou usá-los com componentes que exigem os recursos hierárquicos. Ele implementa também org.w3c.dom.Document, que o torna o Nó de nível superior na hierarquia. Para obter informações adicionais sobre isso, consulte “Trabalhando com Objetos de Entrada Hierárquicos” na página 48.

Existem vários objetos de entrada que são criados e mantidos pelo IBM Security Directory Integrator. A instância mais visível é chamada de *Entrada de trabalho* e serve como a portadora principal de dados em um AssemblyLine (AL). Esse é o depósito utilizado para transportar dados no AL, transmitindo de um componente para o outro.

A Entrada de Trabalho está disponível para uso na criação de scripts por meio da variável pré-registrada *work*, oferecendo acesso direto aos Atributos que estão sendo manipulados por um AssemblyLine (e seus valores). Além disso, todos os Atributos transportados pela Entrada de trabalho são exibidos no Editor de Configuração, no cabeçalho **Atributo de Trabalho** da área Mapas de Atributos na janela Editor de AssemblyLines de um AssemblyLine.

Tipos de Entrada

Há vários objetos de dados que residem em AssemblyLines que seguem o Modelo de dados da entrada. São eles:

Work Essa é a Entrada supramencionada que viaja de um componente para

outro no AssemblyLine e transporta dados entre eles. Seu nome de variável pré-registrada é *work* e está disponível para uso no script em praticamente qualquer lugar[6].

Conn Esse é o objeto semelhante à Entrada que um Conector usa como um intermediário entre o sistema conectado e o AssemblyLine, antes de tornar os dados, ou um subconjunto dele, disponível na Entrada de Trabalho. O processo de mover dados entre Conn e Work é chamado de Mapeamento de Atributos. Seu nome de variável pré-registrado é *conn*, e está disponível para uso no script dentro de vários dos Ganchos em Conectores e Componentes de Função.

Current
Esse objeto semelhante à Entrada está disponível dentro de alguns Ganchos nos Conectores no modo Atualizar e contém os dados do sistema conectado antes de quaisquer atualizações serem aplicadas a ele. Seu nome de variável pré-registrada é *current*.

Error Esse objeto semelhante à Entrada existe apenas em alguns Ganchos em Componentes, quando alguma condição de erro ocorreu e o Gancho de Erro relevante foi chamado. Ele contém informações sobre a exceção real que foi lançada, com possíveis variáveis e dados adicionais, permitindo apontar o que exatamente causou o erro. Seu nome de variável pré-registrado é *error*.

Os diagramas do Fluxo de Conector no *Referência* mostrarão a você quais desses objetos estão disponíveis sob quais circunstâncias.

Consulte Também

"Modelo de Dados Interno: Entradas, Atributos e Valores" na página 46.

6. Quando o AssemblyLine em que você está trabalhando não for chamado com uma Entrada de Trabalho Inicial, então o objeto *work* não estará disponível até *após* os ganchos de Prólogo. Nos ganchos de Prólogo, você tem um código como a seguir:

```
if (work != null) {
 // An Initial work Entry has been provided, we can get values from there
 .... some code
} else {
 // No initial work Entry has been provided
 ... some other code
}
```

Capítulo 2. Script no IBM Security Directory Integrator

O IBM Security Directory Integrator fornece a seus usuários um mecanismo altamente flexível que pode ser customizado a partir dos controles da interface com o usuário do Editor de Configuração, assim como por meio de script de lógica customizada. Enquanto os controles da interface com o usuário fornecem um meio de controlar o fluxo de dados em um nível mais alto, o script fornece a capacidade de controlar quase qualquer aspecto do fluxo de dados, em qualquer nível, incluindo substituir o processamento padrão do IBM Security Directory Integrator. Funções especiais estão disponíveis no objeto *sistema* para reiterar uma entrada do AssemblyLine, ignorar um Conector e iniciar novos AssemblyLines. A linguagem de script usada para implementar essa lógica customizada é JavaScript.

Pronto para uso, o IBM Security Directory Integrator fornece as ferramentas para ajustar rapidamente a estrutura de uma solução de integração. Entretanto, para todas as tarefas de migração, exceto as mais triviais, você precisará customizar e estender o comportamento integrado do produto gravando JavaScript.

O IBM Security Directory Integrator é Java puro. Sempre que emite um comando para o IBM Security Directory Integrator, trabalha com componentes e objetos ou manipula dados em seu fluxo, você está trabalhando com objetos Java. O IBM Security Directory Integrator usa o IBM Java versão 7.0.4.

Por outro lado, sua customização é feita em JavaScript e essa combinação de duas linguagens de programação aparentemente semelhantes, mas essencialmente diferentes, justifica uma verificação mais cuidadosa.

A experiência com o JavaScript será muito útil. Os exemplos fornecidos podem ser incluídos em sua experiência. Entretanto, este manual não ensina o JavaScript propriamente dito, simplesmente seu aplicativo no IBM Security Directory Integrator. Será necessário obter materiais de referência sobre o JavaScript a partir de outras fontes.

Existem diversos guias de referência comercialmente disponíveis sobre o JavaScript, além de documentações, tutoriais e exemplos na Internet. Entretanto, observe que grande parte do conteúdo JavaScript na Web está relacionado ao aprimoramento e à automação do conteúdo HTML. Você precisa se preocupar apenas com a própria linguagem de núcleo, conforme descrito no seguinte link:http://devedge-temp.mozilla.org/library/manuals/2000/javascript/1.5/guide/index.html

Esse site também possui um link muito útil de download de referências no formato HTML, para instalação local. Um excelente manual sobre JavaScript é *The Definitive JavaScript Guide*, 4a. Edição, por David Flanagan (O'Reilly).

Você também desejará os Javadocs para Java, pois todos os objetos do IBM Security Directory Integrator, assim como os valores dos dados dentro de sua solução, estão no formato de objetos Java. Esses documentos estão localizados online nesta URL: http://docs.oracle.com/javase/1.6.0/docs/api/index.html

A própria documentação do J2SE pode ser localizada aqui: http://docs.oracle.com/javase/1.6.0/docs/index.html

O script é necessário quando você precisa incluir processamento customizado em seu AssemblyLine. Exemplos de onde o script pode ser útil incluem as seguintes tarefas:

Manipulação ou cálculo de atributos
Você precisa calcular o valor de um atributo de saída com base em um ou mais atributos de entrada.

Filtragem de dados
Você deseja processar somente entradas que correspondem a um conjunto específico de critérios.

Verificação de validade ou consistência de dados
Você precisa relatar ou corrigir valores de dados inválidos.

Controle de fluxo
Você deseja substituir a operação de atualização do Conector que está sendo usado.

Inicialização
Você deseja executar alguns procedimentos de inicialização antes de seu AssemblyLine ser iniciado.

Cada um dos casos mencionados e vários outros não mencionados requerem geralmente script.

Exemplos

Acesse o subdiretório `examples/scripting` de sua instalação do IBM Security Directory Integrator.

Modelo de Dados Interno: Entradas, Atributos e Valores

Quando os componentes do IBM Security Directory Integrator acessam informações de sistemas conectados, eles convertem os dados de tipos específicos do sistema para uma representação interna usando objetos Java. Na saída, os componentes fazem a conversão contrária, a partir do modelo de dados interno para os tipos nativos do sistema de destino. Essa mesma representação interna é usada quando você deseja transmitir dados para e a partir de AssemblyLines. Portanto, é vital que você entenda como o modelo de dados interno do IBM Security Directory Integrator funciona.

Examinando detalhadamente quando um valor de dados é recebido por um componente, um objeto *Attribute* correspondente do IBM Security Directory Integrator é criado usando o nome do atributo que está sendo lido. O próprio valor de dados (ou valores, no caso de um atributo com diversos valores) é convertido para objetos Java apropriados — como `Java.lang.String` ou `java.sql.Timestamp` — e designado ao Atributo. Se você examinar a documentação da API do IBM Security Directory Integrator, verá que o objeto de Atributo fornece vários métodos úteis, como `getValue()`, `addValue()` e `size()`. Isso permite criar, enumerar e manipular os valores de um Atributo diretamente a partir do script. Você também pode instanciar novos objetos de Atributo, conforme necessário, como mostrado neste exemplo de Mapa de Atributos para mapeamento avançado do atributo objectClass de um diretório:

```
var oc = system.newAttribute( "objectClass" );

oc.addValue( "top" );
oc.addValue( "person" );
oc.addValue( "organizationalPerson" );
oc.addValue( "inetOrgPerson" );

ret.value = oc;
```

Os próprios atributos são coletados em um objeto de armazenamento de dados denominado *objeto de entrada*. A *entrada* é o objeto de portadora de dados principal no sistema e o IBM Security Directory Integrator fornece acesso a objetos de entrada importantes, registrando-os como variáveis do script. Um excelente exemplo é o objeto de entrada de Trabalho no AssemblyLine, usado para transmitir dados entre os componentes do AL (assim como entre os AssemblyLines). Esse objeto de entrada é local para cada AssemblyLine e está disponível como a variável do script *work*.

O IBM Security Directory Integrator fornece alguns atalhos e recursos de conveniência ao trabalhar com JavaScript, portanto, o mapeamento avançado específico acima pode ser codificado simplesmente conforme a seguir:

```
ret.value = [ "top", "person",
"organizationalPerson", "inetOrgPerson" ];
```

O recurso de mapeamento avançado suporta Entradas e matrizes de JavaScript para transmitir múltiplos valores de atributos.

Por exemplo, em um Mapa de Atributos de Entrada (que faz com que os Atributos mapeados sejam mostrados na Entrada *work* no retorno), suponha que você tenha a designação

```
ret.val = anentry;
```

para o Atributo denominado "last". Vamos ainda supor que *work* esteja vazio no início e *anentry* contenha os Atributos "cn", "sn" e "mail".

Após o mapeamento de atributo, *work* conterá os atributos "cn", "sn" e "mail", **não** um único Atributo denominado "last" com "anentry" como valor. Basicamente, o que acontece no mapeamento de atributo é que quando um mapa de atributos retorna um objeto de Entrada, ele é mesclado com a Entrada de recebimento, *work* ou *conn*, dependendo em qual mapa ele se encontra (Entrada ou Saída).

Nota: No IBM Security Directory Integrator, tendo a vantagem de objetos hierárquicos, você pode evitar esse comportamento encapsulando primeiramente uma Entrada em um Atributo antes que o Mapeamento de Atributo ocorra. Por exemplo,

```
// esta é a entrada a ser retornada
e = system.newEntry();
e.setAttribute("some", "value");

// Criar um objeto de Atributo. Não precisamos fornecer um nome porque o mapeamento usará o nome d
attr = system.newAttribute(null);

// incluir a entrada no objeto de Atributo e retornar isso em vez do objeto de Entrada
attr.addValue(e);

return attr;
```

Se isso foi inserido como o Mapa de Atributos Avançado para o Atributo "last", após o Mapeamento de Atributo, a Entrada *work* conterá um Atributo denominado "last". Esse Atributo é uma Entrada que, por sua vez, é constituída de dois atributos chamados "some" e "value".

Examinando os Javadocs, você verá que o objeto de entrada oferece várias funções para trabalhar com Entradas e seus Atributos e valores, incluindo `getAttributeNames()`, `getAttribute()` e `setAttribute()`. Se você desejasse criar e incluir um Atributo na entrada de Trabalho do AssemblyLine, poderia usar o seguinte script, por exemplo, em um Gancho ou em um Componente de Script:

```
var oc = system.newAttribute( "objectClass" );

oc.addValue( "top" );
oc.addValue( "organizationalUnit" )

work.setAttribute( oc );
```

Observe que, nesse caso, **não** existe a opção de usar uma matriz JavaScript para definir o valor:

```
oc.addValue( ["top", "organizationalUnit"] );  // Does not work like Advanced Mapping
```

Esse código resultará no atributo `oc` obtendo um único valor, que por sua vez é uma matriz de cadeias.

Objetos de Entrada também podem conter *propriedades*. As propriedades são contêineres de dados, como Atributos, exceto que elas são somente de valor único. Enquanto Atributos são usados para armazenar o conteúdo de dados, propriedades contém informações paramétricas, possibilitando que essas informações permaneçam separadas. As propriedades não são mostradas para a seleção do mapa de atributos ou na lista de entradas de Trabalho, mas podem ser acessadas do mesmo modo que os Atributos a partir do script. As funções de entrada como `getProperty()` e `setProperty()` são usadas para isso e funcionam diretamente com os valores das Propriedades, que podem ser qualquer tipo de objeto Java, exatamente como os valores de Atributos. Não há nenhum objeto de Propriedade intermediário, tal como existe quando você trabalha com Atributos.

Em muitos casos, você pode restringir o modelo de dados a uma entrada contendo zero ou mais Atributos, cada um com zero ou mais valores—um esquema simples.

Esta é uma das eficácias do IBM Security Directory Integrator: simplificar e harmonizar representações de dados e esquema. Isso também impõe desafios quando se trata de manipular informações com uma estrutura mais complexa. Entretanto, como o valor de Atributo pode ser qualquer tipo de objeto Java, incluindo um outro objeto de entrada (com seus próprios Atributos e valores), o IBM Security Directory Integrator permite trabalhar com dados estruturados hierarquicamente.

Essa maneira mais elaborada e estruturada de manipular objetos hierárquicos é descrita em “Trabalhando com Objetos de Entrada Hierárquicos”.

Trabalhando com Objetos de Entrada Hierárquicos

Uma maneira alternativa de trabalhar com dados hierarquicamente estruturados é aproveitar o suporte para objetos hierárquicos no objeto de entrada do IBM Security Directory Integrator.

Diferente de versões anteriores, o IBM Security Directory Integrator Versão 7.1.1 e mais recente suporta o conceito do objeto de Entrada hierárquico. O objeto de Entrada representa a raiz da hierarquia e cada Atributo representa um nó nessa hierarquia. Seguindo esta lógica, os valores de cada Atributo são folhas na hierarquia. Uma API para atravessar a hierarquia também existe. Essa API é uma implementação parcial da especificação DOM 3. Somente algumas classes dessa especificação foram implementadas:

- `Org.w3c.dom.Document` – implementada pela classe de Entrada.
- `Org.w3c.Element` – implementada pela classe de Atributo.
- `Org.w3c.Attr` – implementada pela classe de Propriedade.
- `Org.w3c.Text` e `org.w3c.CDATASection` – implementadas pela classe de AttributeValue.

Essa classes são o conjunto mínimo de classes fornecidas pela especificação DOM que são necessárias para representar os dados hierárquicos. A API pré-v7.0 não reconhece hierarquia (por exemplo, não pode acessar/modificar/remover Elementos-filho) por razões de compatibilidade com versões anteriores. Esse é o motivo pelo qual somente a API do DOM pode manipular uma estrutura hierárquica.

Para manter a estrutura de Entradas compatível com versões anteriores, por padrão, a Entrada sempre usa Atributos simples. A Entrada torna-se hierárquica somente on demand – após você chamar uma das APIs do DOM recém-fornecidas. Isso permite que somente componentes que reconheçam a natureza hierárquica da Entrada a usem, o restante dos componentes não precisa ser alterado a fim de mantê-los em execução. Iniciando a partir do IBM Security Directory Integrator v7.0, uma nova notação de nome foi introduzida para que os usuários tenham uma maneira fácil de criar árvores hierárquicas. Todo nome contendo um ponto é considerado um nome composto constituído de nomes simples; esses nomes são separados por pontos. Quando um nome composto é passado para uma Entrada hierárquica, ela o fragmenta em nomes simples e constrói a hierarquia descrita por esse nome composto.

Por exemplo, se você executar o seguinte código JavaScript:

```
// criar um novo objeto de Entrada vazio
var entry = new com.ibm.di.entry.Entry(true);
// criar uma nova ramificação de 2 níveis
entry.setAttribute("firstLevelChild.secondLevelChild", "level2Value");
// localiza a ramificação já existente e cria um novo nó no nível 3
entry.setAttribute("firstLevelChild.secondLevelChild.thirdLevelChild", "level3Value");
```

a seguinte estrutura será criada:

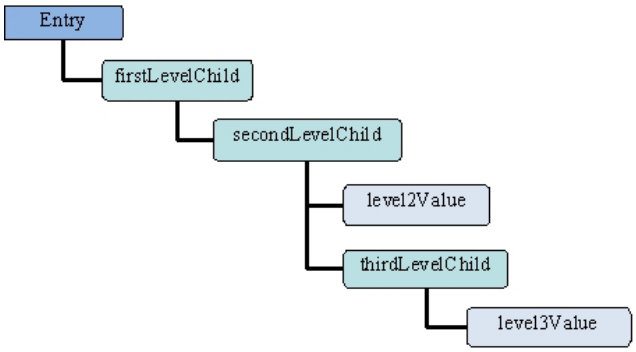


Figura 1. Entrada Hierárquica Simples

Nota: É importante saber que os nomes não são fragmentados em nomes simples enquanto a estrutura de Entrada não for convertida em hierárquica. Por exemplo, se a entrada fosse simples e somente os métodos antigos fossem usados, isso ainda criaria a mesma estrutura simples antiga:

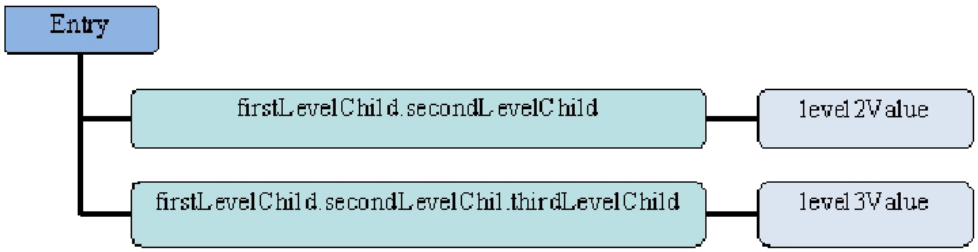


Figura 2. Tradicional, Entrada Simples

Em alguns casos, pode ser necessário ter pontos nos nomes dos atributos da Entrada, portanto, o objeto de Entrada também reconhece caracteres de escape (atualmente, somente \\ e \. são suportados).

Nota: Ao trabalhar a partir do script, deve-se fazer o escape da barra invertida com uma outra barra invertida!
Por exemplo, se a seguinte hierarquia for necessária:

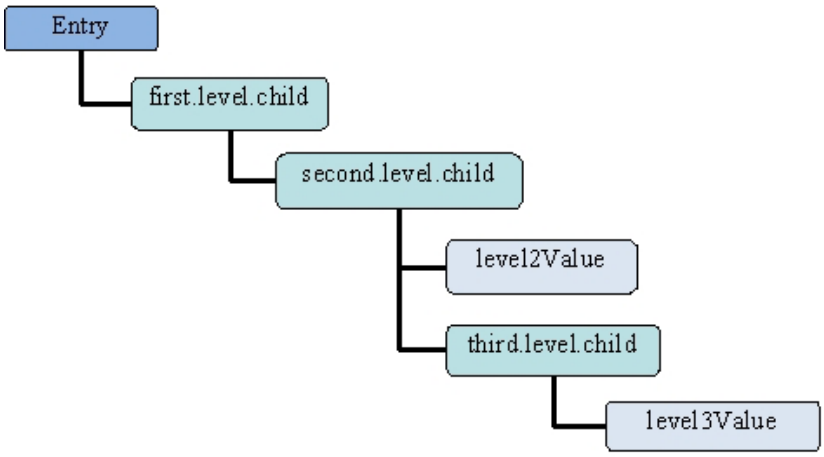


Figura 3. Outra Entrada Hierárquica Simples

O seguinte script a criará:

```
var entry = new com.ibm.di.entry.Entry(true);
entry.setAttribute("first\\.level\\.child.second\\.level\\.child", "level2Value");
entry.setAttribute("first\\.level\\.child.second\\.level\\.child.third\\.level\\.child", "level3Value");
```

Para manter a compatibilidade com releases anteriores do IBM Security Directory Integrator ao trabalhar implicitamente com Entradas hierárquicas, todos os métodos antigos expostos pelas classes de Atributo e de Entrada foram alterados um pouco. Por exemplo, o método `Entry.getAttributeNames()` retornará uma matriz de caminhos completos para os filhos que estão disponíveis na árvore. Em referência à estrutura acima, o método getAttributeNames retorna a matriz:

```
["first\.level\.child.second\.level\.child", "first\.level\.child.second\.level\.child.third\.level\.child"]
```

O método `Entry.size()` retorna o número total de elementos na matriz retornados pelo método `getAttributeNames()`. Em nosso caso, o método size() retornaria

return 2 (o número de folhas na árvore inteira) em vez de 1 (como você poderia esperar, considerando que o objeto de Entrada possui somente um atributo como filho).

Isso ocorre porque os métodos getAttributeNames() e size() trabalham somente com estruturas simples. Para obter o tamanho real dos filhos, você precisaria usar a API do DOM conforme a seguir: `Entry.getChildNodes().getLength();`

Se a entrada for simples, a API antiga se comportará como em releases anteriores.

O Objeto de Atributo

O objeto de Atributo foi aprimorado com a capacidade para criar estruturas hierárquicas. Essas estruturas seguem a especificação DOM e esse é o motivo pelo qual o objeto de Atributo reconhece os conceitos de Espaço para Nome XML.

A classe de Atributo também precisou ser expandida para fornecer novos métodos para a funcionalidade hierárquica. Os métodos getValue/setValue/addValue também são compatíveis com versões anteriores e retornarão somente os valores que o elemento específico possui. A diferença aqui é que quando cada valor for acessado por meio da API do DOM, ele será quebrado em uma classe AttributeValue e será tratado como um nó Texto. Para obter os elementos-filho de um Atributo (por exemplo, Atributos e AttributeValues) a API do DOM precisa ser usada.

Um filho de Atributo de uma Entrada também estará apto a comutar a estrutura da Entrada para hierárquica quando qualquer um de seus métodos DOM for acessado. Diferente da classe de Entrada, a classe de Atributo faz isso implicitamente e não fornece uma maneira de fazer a comutação explicitamente.

O IBM Security Directory Integrator v7.0 suporta extensões para recursos de script do Servidor para acessar facilmente as estruturas complexas. Consulte a seção “Navegação em Scripts” na página 52 para obter detalhes adicionais.

O Objeto de AttributeValue

Essa classe representa um valor na árvore hierárquica. De acordo com a especificação DOM, os valores na árvore são sempre Cadeias. Entretanto, a classe de AttributeValue, possui objetos retidos de qualquer tipo para os últimos releases. Isso é válido também na versão atual. A única diferença para o objeto de AttributeValue é que, quando acessado por meio do DOM, ele retornará uma representação de Cadeia do objeto contido. Para que a classe de AttributeValue represente um Nó e tenha um valor ao mesmo tempo nos termos que a especificação DOM define, ela deve implementar as interfaces org.w3c.dom.Text ou org.w3c.dom.CDATASection. O AttributeValue implementa ambas e pode representar qualquer um desses Nós, dependendo de suas necessidades.

O Objeto de Propriedade

Os atributos podem ter zero, um ou mais objetos de Propriedade. A classe de Propriedade implementa a interface org.w3c.dom.Attr e, portanto, representa os atributos em termos de conceitos do DOM. Usando as propriedades, você pode declarar prefixos/espaços de nomes em termos de conceitos de XML.

Transferindo Objetos

Mapear um atributo de uma entrada para outra sempre copiará o Atributo de origem.

Por exemplo:

```
entry.appendChild(conn.getFirstChild());
// ou
entry.setAttribute("name", conn.getFirstChild());
```

Mesmo quando o Atributo não é um filho de primeiro nível da Entrada, ainda assim ele é copiado. Isso também pode ser feito pelo script:

```
entry.a.b.c.d.appendChild(conn.e.f.g);
```

Para mover um objeto de Atributo entre as entradas sem cloná-lo, você precisará primeiro desanexá-lo de seu pai antigo, depois, anexá-lo a um novo pai.

Por exemplo:

```
var src = entry1.b.source;
entry1.b.removeChild(src);
entry2.a.target.appendChild(src);
```

Ao mover um objeto de Atributo de um pai para outro, na mesma Entrada, o Atributo é movido automaticamente. Não é feito nenhum clone.

Por exemplo:

```
entry.a.target.appendChild(entry.b.source);
```

Neste exemplo, o Atributo "source" é desanexado de seu pai ("entry.b") e, então, anexado ao Atributo "entry.a.target". Não é feito nenhum clone.

Se não desejar remover o objeto de Atributo da origem, você poderá anexar uma cópia do Atributo como esta:

```
entry.a.target.appendChild(entry.b.source.clone());
```

Navegação em Scripts

O ScriptEngine do IBM Security Directory Integrator possibilita acessar facilmente os atributos de uma entrada simplesmente fazendo referência a eles pelo nome; por exemplo, `entry.attrName` retorna o atributo com o nome *attrName*.

1. O Mecanismo de JavaScript resolverá nomes com base no objeto de contexto no qual o nome foi solicitado. Por exemplo, se a chamada `entry.a` for executada, o nome *entry* será o objeto de contexto e *a* será o nome do objeto-filho a ser resolvido. O Mecanismo de JavaScript utiliza interpretação da esquerda para a direita para avaliar cada objeto de contexto até o final ser resolvido. Com base no diagrama abaixo, a seguinte chamada, `entry.a.b.c`, é resolvida usando este procedimento: Localize o objeto *entry* para usá-lo como o objeto de contexto para a primeira etapa.
2. Procure o objeto de contexto para um nome *a*. O objeto *entry* possui somente um filho com nome *a*. Considere esse filho como o próximo objeto de contexto para a próxima etapa.
3. Procure o objeto de contexto para um nome *b*. O objeto de contexto possui dois filhos nomeados *b*. Coloque-os em uma lista e retorne essa lista.

4. A operação final procura na lista retornada na operação anterior pelo nome *c*. Cada elemento na lista possui pelo menos um filho. Obtenha todos eles e coloque-os em uma lista, que é o resultado real da resolução da expressão integral.

O diagrama de exemplo de objeto de entrada a seguir ilustra isso:

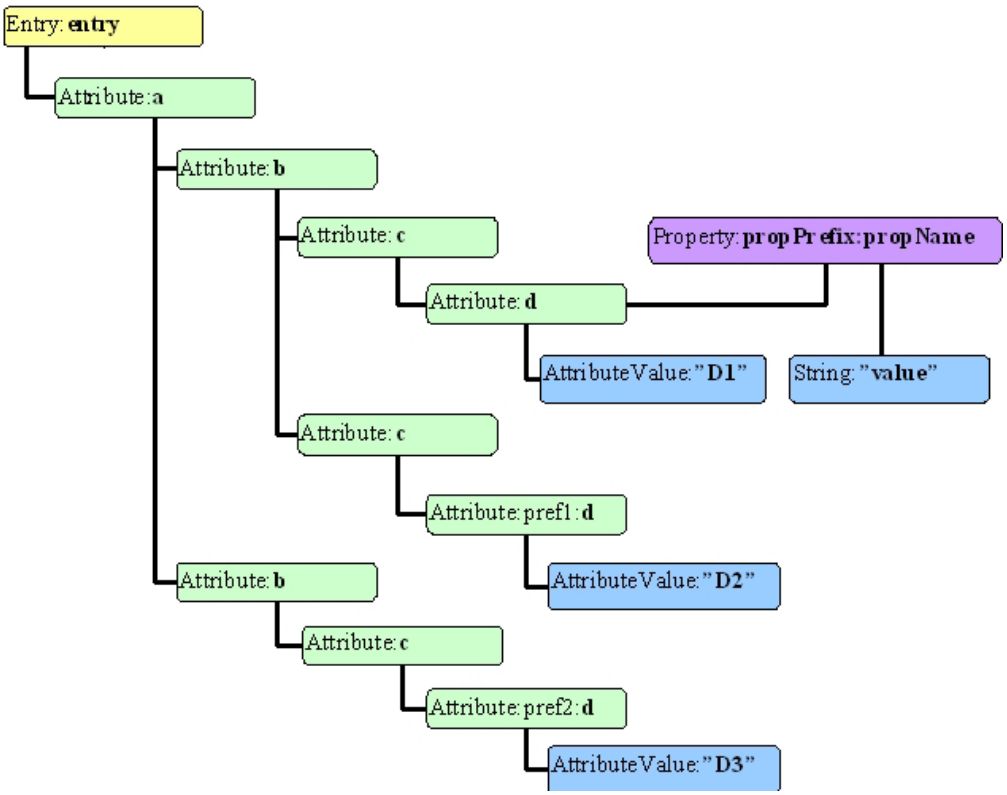


Figura 4. Exemplo de Objeto de Entrada Hierárquica

O mecanismo de script fornecido pelo IBM Security Directory Integrator permite que mais nomes arbitrários sejam usados no processo de resolução de filhos. Por exemplo, se o nome do filho contiver pontos, você poderá se referir a ele utilizando a sintaxe de colchetes conforme mostrado a seguir:

```
work["{namespace}name:Containing\.invalid\.charaters"]
```

Observe que os pontos estão sendo escapados para denotar que fazem parte do nome local e que não devem ser tratados como separadores de caminho pelo mecanismo de script.

Dependendo do objeto de contexto atual no qual uma operação é executada, o resultado final pode ser diferente. O IBM Security Directory Integrator adere à manipulação de objeto padrão fornecida pelo Mecanismo de JavaScript através da implementação de vários aprimoramentos nos seguintes objetos:

Entrada

Quando o objeto de contexto for uma instância desse tipo, o mecanismo de resolução de nome procurará a seguinte sintaxe:

- `@<nome>` – procura o objeto de Entrada para uma propriedade com o nome especificado. O objeto resolvido é nulo ou o objeto mapeado para essa propriedade.

- `<prefixo>:<localName>` – procura o objeto de Entrada para um filho que possui um prefixo igual a <prefixo> e um Nome Local igual a <localName>. O objeto resolvido pode ser nulo ou um objeto de Atributo existente.
- `<localName>` – procura o objeto de Entrada para o primeiro filho que possui um Nome Local igual a <localName>. O objeto resolvido pode ser nulo ou um objeto de Atributo.
- `{namespaceURI}<localName>` – procura a Entrada para o primeiro Atributo filho que pertence ao namespaceURI especificado e possui o mesmo Nome Local que o nome especificado.

  Nota: Se um prefixo for fornecido, ele será ignorado e o mecanismo de resolução de nome procurará somente o namespaceURI e o localName especificados. O objeto resolvido pode ser nulo ou um objeto de Atributo.

Atributo

Quando o objeto de contexto for uma instância desse tipo, o mecanismo de resolução de nome procurará a seguinte sintaxe:

- `@<prefixo>:<localName>` e `@<localName>` – procura o Atributo para objetos de Propriedade que possuem o mesmo prefixo e/ou nome local ou apenas o nome local especificado. Retornará nulo, se nenhuma propriedade corresponder ao nome especificado, ou um único objeto de Propriedade.
- `@{namespaceURI}<localName>` – procura o Atributo para uma Propriedade que pertence ao namespaceURI especificado e possui o mesmo Nome Local que o nome especificado.

  Nota: Se um prefixo for fornecido, ele será ignorado e o mecanismo de resolução de nome procurará somente o namespaceURI e o localName especificados. O objeto resolvido pode ser nulo ou um objeto de Propriedade.
- `[<índice>]` – especifica a posição do valor no Atributo a ser recuperado. Usando essa notação, não é possível acessar um filho desse Atributo. Retorna nulo ou o Objeto na posição especificada.
- `<prefixo>:<localName>` e `<localName>` – procura o Atributo para um filho ou filhos com o prefixo e/ou nome local especificados. Retorna nulo ou o filho com o nome especificado (se somente um), ou uma NodeList com todos os Atributos-filho que correspondem aos critérios.
- `{namespaceURI}<localName>` – procura o Atributo para todos os Atributos-filho que pertencem ao namespaceURI especificado e possuem o mesmo Nome Local que o nome especificado.

  Nota: Se um prefixo for fornecido, ele será ignorado e o mecanismo de resolução de nome procurará somente o namespaceURI e o localName especificados. O objeto resolvido pode ser nulo, um único objeto de Atributo ou uma NodeList contendo todos os Atributos que correspondem ao nome de procura.

NodeList

Quando o objeto de contexto for uma instância desse tipo, o mecanismo de resolução de nome procurará a seguinte sintaxe:

- `[<índice>]` – especifica a posição do elemento na NodeList a ser recuperada. Retorna nulo ou o Objeto na posição especificada. Lançará uma exceção se o índice estiver fora dos limites.

- `@<prefixo>:<localName>` e `@<localName>` – procura cada um dos elementos da NodeList para uma propriedade que possui o mesmo prefixo e/ou nome local. Retorna nulo, um objeto de Propriedade (se somente um for localizado) ou uma Lista de todos os objetos de Propriedade localizados na NodeList.
- `@{namespaceURI}<localName>` – procura cada um dos Atributos para uma Propriedade que pertence ao namespaceURI especificado e possui o mesmo Nome Local que o nome especificado.

  Nota: Se um prefixo for fornecido, ele será ignorado e o mecanismo de resolução de nome procurará somente o namespaceURI e o localName especificados. O objeto resolvido pode ser nulo ou um objeto de Propriedade (se somente um for localizado) ou uma NodeList de todos os objetos de Propriedade localizados na NodeList.
- `<prefixo>:<localName>` e `<localName>` – procura cada um dos elementos da NodeList para um filho que possui o mesmo prefixo e/ou nome local. Retorna nulo, um objeto de Atributo (se somente um for localizado) ou uma NodeList de todos os objetos de Atributo localizados na NodeList.
- `{namespaceURI}<localName>` – procura cada um dos Atributos para todos os Atributos-filho que pertencem ao namespaceURI especificado e possuem o mesmo Nome Local que o nome especificado.

  Nota: Se um prefixo for fornecido, ele será ignorado e o mecanismo de resolução de nome procurará somente o namespaceURI e o localName especificados. O objeto resolvido pode ser nulo, um único objeto de Atributo ou uma NodeList contendo todos os Atributos que correspondem ao nome de procura.

Agora você tem as seguintes opções:

1. Capacidade para acessar todos os elementos *d*, fazendo referência a eles iniciando da parte superior; por exemplo, `entry.a.b.c.d` – isso retorna um objeto do tipo NodeList com todos os atributos *d* que correspondem a esse caminho. Em nosso exemplo, isso retornará todos os três elementos *d*.
2. Capacidade de acessar atributos especificando prefixo e nome local; por exemplo, `entry["a.b.c.pref1:d"]` – isso retornará um único Atributo, aquele contendo um prefixo "pref1".
3. Capacidade para acessar cada Atributo de uma NodeList usando a notação [ ], por exemplo, `entry.a.b.c.d[0]` – isso retorna um único Atributo, a saber, o primeiro elemento *d* da estrutura acima.
4. Capacidade para navegar através dos elementos usando a notação [ ]; por exemplo, `entry.a.b[0].c.d` – isso retorna uma Lista de Atributos, mas desta vez, contém todos os atributos *d* da primeira ramificação *b*.
5. Capacidade para obter a propriedade de um atributo (ou seja, um Atributo de um Elemento usando a convenção de nomenclatura do DOM) usando a notação @; por exemplo, `entry.a.b.c.d[0]["@propPrefix:propName"]` – isso retorna um objeto de Cadeia contendo o valor dessa propriedade (ou seja, o valor do Atributo de acordo com o DOM). Observe aqui que o propPrefix e o propName estão separados pelo sinal de dois pontos (":"), e se a propriedade tiver um prefixo, será obrigatório especificar o prefixo e o nome para que a propriedade seja localizada. Alternativamente, o espaço de nomes e o propertyName podem ser usados para localizar uma propriedade que possui um prefixo.

6. Capacidade para chamar métodos de um Atributo antes de procurar o filho com o nome do método que o usuário deseja executar; por exemplo, `entry.a.b.[0].getChildNodes()` – isso retorna um objeto de NodeList que possui todos os valores de Atributo que o primeiro Atributo *b* contém.
7. Capacidade para acessar atributos de entrada por nome; por exemplo, `entry.attrName` – isso retornará o Atributo mapeado para a chave *attrName*.
8. Capacidade para acessar as propriedades de entrada usando a notação .@; por exemplo, `entry.@propName` – isso retornará o Objeto mapeado como propriedade para a chave *propName*.
9. Capacidade para acessar nós-filho especificando o espaço de nomes ao qual esses filhos pertencem. Por exemplo, `work.a.b[{someNamespace}c]` – isso retornará todos os objetos *c* que pertencerem ao espaço de nomes "someNamespace".

Nota:

1. Se o nome de um atributo for igual ao nome de um método de uma Entrada/Atributo, o método será chamado. Se você desejar acessar o atributo, deverá usar os métodos do objeto como antes (ou seja, `Entry.getAttribute("getAttribute");`.)
2. Se uma entrada simples for usada, o mecanismo de script não converterá a entrada para hierárquica, a menos que o espaço de nomes do atributo a ser localizado seja especificado. Por exemplo: `entry["{ns}element"]`. O mecanismo de script converterá automaticamente a entrada para hierárquica se o objeto de contexto for do tipo Atributo. Por exemplo, neste caso: `entry.attr.child`.
3. Se o atributo a ser localizado contiver pontos em seu nome e a entrada for simples, você deverá usar a notação de colchete em vez da notação de ponto ou forçar a entrada a tornar-se hierárquica antes de resolver o nó-filho. Por exemplo, se a entrada for simples, você não poderá acessar o atributo `http.body` usando a chamada `entry.http.body`. Para que isso aconteça, você precisará usar isto no lugar: `entry["http.body"]` ou chamar `entry.enableDOM()` antes de chamar `entry.http.body`.
4. Se você pretender usar a referência recuperada da expressão, por exemplo, `a.b.c.d`, mais de uma vez, será uma boa prática designar essa referência a uma variável local, pois cada avaliação de repetição da mesma expressão resultará em sobrecarga adicional para recuperar a mesma referência.
5. Se, por exemplo, a expressão `entry.a.b.c[0].d` for usada, isso fará referência a um objeto de Atributo; mas se `entry.a.b[0].c.d` for usada, isso fará referência a um objeto de NodeList. Para reconhecer o objeto de referência, você pode verificar o nome do objeto, por exemplo:

```
var obj = a.b.c.d;
if (obj.getClass().getSimpleName().equals("Attribute") ) {
  // tratar isto como Atributo
} else {
  // tratar isto como uma lista de Atributos
}
```

Se, por exemplo, você precisar tratar de cada elemento retornado por uma expressão, mesmo se somente um elemento for retornado, uma estrutura de loop for/in como esta poderá ser usada:

```
for (obj in entry.a.b.c.d) {
  // o obj será um objeto do tipo Atributo
}
```

O mesmo uso está agora disponível no objeto de Entrada, por exemplo

```
for (obj in work) {
  // o obj será um atributo de Entrada
}
```

6. ScriptEngineOptions também permite a designação de valores. Por exemplo, as seguintes expressões são válidas:

   entry.a.b.c.d[0]["@propPrefix:propName"] = "novo valor";
   Isso altera o valor da propriedade. Se a propriedade não existir, ela será criada.

   entry.a.b.c.d[0][0] = "novo valor";
   Isso substitui o valor de atributo na posição 0.

   entry.a.b.c.d[0][1] = "um outro valor";
   Isso inclui um outro valor no atributo (se o índice for igual ao tamanho da matriz de valores.)

   entry.a.b.c.d[0][a.b.c.d[0].size()] = "valor anexado";
   Se precisar anexar um novo valor à lista de objetos, mas não souber a posição do último elemento, você poderá utilizar `Attribute#size()` para obter o número de filhos que esse elemento tem.

   new com.ibm.di.entry.Entry().@propName = "someValue";
   Isso criará uma nova propriedade no objeto de Entrada com o nome propName, se não existir, e configurará seu valor para a Cadeia "someValue".

   entry.a.b.c[1] = "valor";
   Isso resolverá *entry.a.b.c* como uma NodeList e incluirá automaticamente como um valor o novo objeto de Cadeia no segundo elemento do objeto NodeList resolvido. Se, por exemplo, *entry.a.b.c* for resolvido para um Atributo, o novo valor de Cadeia substituirá o segundo valor do Atributo *c* resolvido.

   entry.a.b.c[2]["pref3:d"] = "valor";
   Isso incluirá um novo filho *pref3:d* no atributo *c* a partir da lista *entry.a.b.c*. Observe que *entry.a.b.c[2]* é resolvido para um Atributo e para uma lista.

   entry.a.b.c["{namespaceURI}:d"] = "myTextValue";
   Isso configura um valor para o primeiro Atributo-filho de *c* que possui um nome local igual a *d* e pertence ao espaço de nomes `namespaceURI`. Se nenhum atributo desse tipo for localizado, um novo será criado e o valor será designado a ele. Ao criar um atributo, você também pode associar o espaço de nomes a um prefixo. Em nosso caso, isso poderia ser feito com isto: `entry.a.b.c["{namespaceURI}prefix:d"] = "myTextValue";`.

   entry["{http://ibm.com/xmlns/}first.second.third"] = null;
   Isso tentará localizar primeiramente o elemento com o nome local "first" no namespace "http://ibm.com/xmlns/". Quando falhar, tentará criar o elemento e depois resolver seu elemento filho "second". Quando falhar, tentará criar e configurar seu espaço de nomes para o primeiro elemento. Por fim, será feita uma tentativa de resolver o terceiro elemento. Quando falhar, o último elemento será criado sem nenhum valor. Isso é equivalente a `entry["{http://ibm.com/xmlns/}first.{http://ibm.com/xmlns/}second.{http://ibm.com/xmlns/}third"] = null;`

Criando a Estrutura Acima com Script

```
// criar uma nova entrada que conterá a estrutura
var entry = new com.ibm.di.entry.Entry();
// criar a primeira ramificação
var d = entry.newAttribute("a.b.c.d");
// criar uma nova propriedade e designar um valor a ela
d["@propPrefix:PropName"] = "valor";
// criar um novo valor do Atributo d
d[0] = "D1";
// anexar um novo valor para o atributo entry.a.b
entry.a.b.appendChild(entry.createElement("c"));
// criar um novo Atributo d com um prefixo no segundo atributo c
// e designar a cadeia "D2" como um primeiro valor
entry.a.b.c[1]["pref1:d"] = "D2";
// criar um novo filho do Atributo a
entry.a.appendChild(entry.createElement("b"));
// escolher o segundo atributo a partir da NodeList entry.a.b e criar um novo filho nomeado c
entry.a.b[1].appendChild(entry.createElement("c"));
// criar o filho d do Atributo c
entry.a.b[1].c["pref2:d"] = "D3";
```

Navegação Usando XPath

A classe de Entrada fornece métodos convenientes para navegar e recuperar dados com base em expressões XPath. Usar XPath para consultar dados a partir de uma Entrada é muito mais avançado do que usar qualquer navegação simples dentro de scripts. É muito mais fácil implementar uma procura e/ou uma lógica de correspondência de valor em uma única expressão do que gravar scripts multilinhas para atingir os mesmos resultados.

A classe de Entrada fornece os seguintes métodos:

- `NodeList getNodeList (String xPath);`
- `Attribute getFirstAttribute(String xPath);`
- `String getStringValue(String xPath);`
- `Number getNumberValue(String xPath);`
- `Boolean getBooleanValue(String xPath);`

Nivelando as Estruturas Hierárquicas

Ao trabalhar com dados hierárquicos, às vezes é necessário nivelar os nós da hierarquia. Para fazer isso, você pode gravar um script para atravessar a árvore ou usar um dos métodos:

- `getElementsByTagName(String namespace, String localName);`
- `getElementsByTagName(String tagName);`

Esses métodos atravessam a árvore e procuram elementos com o nome e o espaço de nomes especificado. A API do DOM permite que esses métodos aceitem o caractere "*" para nomes e espaços de nomes. Esse caractere representa um curinga, que é usado para corresponder qualquer nome/espaço de nomes de elemento O uso desse caractere para nome nivela a árvore retornando todos os nós de Atributos em uma NodeList. Deve-se observar que a estrutura da hierarquia não foi alterada. A NodeList retornada é simplesmente um contêiner para o nós de Elementos que foram localizados na árvore.

Se você executar o seguinte script na entrada a partir da estrutura na seção “Navegação em Scripts” na página 52:

```
var list = entry.getElementsByTagName("*");
```

a variável *list* conterá a seguinte estrutura:

```
list
   |
  + a
   |
  + b
```

```
 |
+ c
 |
+ d
 |
+ c
 |
+ pref:d
 |
+ b
 |
+ c
 |
+ pref2:d
```

Exceções

Uma exceção será lançada nos seguintes casos:

- Se o método `entry.appendChild(newAttr)` for chamado e a entrada já contiver um Atributo com o nome do objeto newAttr passado para o método.
- Se, para qualquer um dos métodos definidos pelo Documento/Nó/Elemento e implementados pelas classes de Entrada/Atributo, um parâmetro inesperado for passado.
- Uma ArrayIndexOutOfBoundsException será lançada se o script fornecido fizer referência a um índice de um Atributo/NodeList que não existe.

Integrando Scripts na sua Solução

Como já foi explicado, você usa script onde quer que precise de um processamento customizado na sua solução de integração. As melhores práticas com o IBM Security Directory Integrator dividem esse processamento customizado em duas categorias: transformação de atributo e controle de fluxo.

Nota: Esta é a convenção e não uma limitação ou regra impostas pelo sistema. A necessidade de customizar o processamento de dados inevitavelmente vem de algum ponto identificável no fluxo de dados (por exemplo, antes que qualquer processamento seja inicializado, antes do processamento de uma determinada entrada, depois de uma falha e assim por diante, então, colocando seu código como fechado neste ponto da forma que for possível, você pode fazer soluções que são mais fáceis de entender e de fazer manutenção.

O local lógico para fazer as transformações do atributo é no **Mapas de Atributos**, tanto Entrada como Saída. Se precisar computar um novo atributo que seja requerido pela lógica do script ou outro downstream dos Conectores no AssemblyLine, isso é o melhor a fazer em um **Mapa de Entrada** se possível. Ou então, se você tiver de transformar atributos por causa de uma única origem de saída, poderá evitar a mistura do objeto de entrada de **trabalho** com transformações específicas de saída, colocando-as no **Mapa de Saída** do Conector relevante.

A outra categoria de lógica de customização, controle de fluxo, é melhor implementada nos Ganchos que são chamados naquele ponto no Workflow automatizado onde a lógica é necessária. Esses pontos de controle são facilmente acessados a partir do Editor de Configuração. A implementação do processamento customizado é simplesmente uma questão de identificar o ponto de controle correto e inserir seu script na janela de edição apropriada.

Os **Componentes do Script** do AssemblyLine, blocos independentes de código com script, também fornecem a você um local para criar seu próprio processamento customizado e, em seguida, permite que você reposicione seu código dentro do AssemblyLine. Embora os Componentes de Script sejam frequentemente usados durante o teste e a depuração, eles também podem desempenhar uma importante função em um Config de produção. Lembre-se apenas de nomear os componentes com clareza e incluir alguma documentação no próprio script para explicar [7] porque você implementou essa lógica em um Componente do Script e não em um **Mapa de Atributos** ou um **Gancho**.

Embora seja importante que os dois identifiquem corretamente o ponto de controle apropriado em que você entrou o seu script, é igualmente importante limitar o escopo do seu script para abranger apenas o objetivo único associado ao ponto de controle. Se você mantiver suas unidades de lógica independentes entre si, então haverá uma chance maior de que elas sejam reutilizáveis e uma chance menor de que possam ser interrompidas quando os componentes forem reordenados ou reutilizados em outros contextos. Uma forma de construir código reutilizável é criar suas próprias funções na sua **Biblioteca de Script** (ou em um Gancho **Prólogo**) para implementar frequentemente a lógica usada, em vez de copiar e colar o mesmo código em vários locais.

Para somar algumas das melhores práticas que você deseja ter em mente enquanto constrói as soluções:

- Manipule o atributo em Mapas de Atributos.
- Coloque o controle de fluxo (filtragem, validação, ramificação e assim por diante) em Ganchos e, onde necessário, os componentes de script do AssemblyLine.
- Use o comportamento automatizado sempre que possível; por exemplo, o fluxo de trabalho do AssemblyLine e os modos do Conector.
- Simplifique sua solução mantendo os AssemblyLines curtos e focados.
- Geralmente, coloque a lógica usada em blocos separados; por exemplo, Scripts em sua seção Recursos.
- Pense na reutilização.

Vale a pena mencionar novamente que, embora os métodos explicados anteriormente sejam o melhor a fazer, você pode encontrar situações onde terá que desviar da convenção estabelecida. Se for este o caso, a documentação da sua solução é vital para fazer a manutenção e aprimorar seu trabalho com o tempo.

Controlando a Execução com Scripts

O mecanismo expõe várias classes e objetos que podem ser acessados, lidos e modificados a partir de scripts criados pelo usuário em um AssemblyLine. Esses objetos representam o estado do AssemblyLine e o ambiente inteiro do IBM Security Directory Integrator a qualquer momento. Ao modificar qualquer um desses objetos, você modifica o ambiente do IBM Security Directory Integrator e, então, afeta a execução do processo de integração.

Nota: Alterações podem ser aplicadas às instâncias de um componente ou um AssemblyLine. Alterações podem ser feitas também nos parâmetros operacionais,

7. Aos outros, bem como a você mesmo quando é preciso revisitar o seu código algum tempo depois!

como no sistema ou nos parâmetros Java. Alterações podem ser feitas também no arquivo de configuração ou no Config. Nesse caso, novas instâncias de objetos Config refletem essas alterações.

Para obter mais informações sobre objetos globais, consulte os Javadocs incluídos como parte do produto IBM Security Directory Integrator selecionando **Ajuda** > **Javadocs** no Editor de Configuração.

Uma descrição de todas as classes e instâncias disponíveis pode ser encontrada no pacote de instalação.

Compreendendo as classes e interfaces expostas, você pode entender melhor os elementos do IBM Security Directory Integrator, bem como as relações entre eles.

Usando Variáveis

É importante distinguir entre o objeto do contêiner padrão para dados sendo processados (o objeto Entrada) e outras variáveis e tipos de dados genéricos fornecidos a você por JavaScript, bem como aqueles que você mesmo cria. Sua criatividade e os recursos da linguagem de script são suas únicas restrições em termos do que pode ser colocado nos scripts dentro de suas soluções do IBM Security Directory Integrator. Entretanto, ao manipular dados no contexto do fluxo de dados, você deve estar ciente da estrutura do objeto de Entrada e usá-la.

Os objetos de Entrada carregam atributos que são o próprio contêiner para valores de dados. Os valores de atributos são eles mesmo objetos (`java.lang.String`, `java.util.Date` e estruturas mais complexas). Um valor de atributo pode até mesmo ser outro objeto de entrada com seu próprio conjunto de atributos e valores. É tarefa do IBM Security Directory Integrator compreender como os dados são armazenados no sistema conectado, como também como converter esses tipos nativos de e para a representação de dados do sistema, que está em objetos Java.

Se você conhece a classe do valor de atributo, poderá acessar e interpretar com êxito esse valor. Por exemplo, se um atributo `java.lang.String` contiver um valor de ponto flutuante que você deseja usar como um ponto flutuante, você deve primeiro transformar manualmente esse valor, por meio da linguagem de script, para algum tipo de dado numérico.

Ao criar variáveis ou processos não diretamente relacionados à flutuação de dados do processo de integração e à disponibilidade dos objetos globais, aplicam-se os seguintes princípios: você pode declarar e usar qualquer variável (objetos) ativados pela linguagem de script. A finalidade dessas variáveis é ajudá-lo a atingir o objetivo específico associado ao ponto de controle no qual você fez o script. As variáveis devem servir somente como buffers temporários e não tentar afetar o estado do ambiente do IBM Security Directory Integrator.

Usando Propriedades

Durante o tempo de vida do AssemblyLine, o Servidor IBM Security Directory Integrator disponibiliza várias Propriedades do componente, relacionadas ao ambiente de execução do AssemblyLine, que podem ser consultadas nos scripts: nos Componentes do Script ou nos Ganchos dos Componentes. Uma propriedade (*lastCallStatus*) pode até mesmo ser definida.

Você acessa as propriedades usando um objeto AssemblyLine, por exemplo, um Conector e chama seu método `get(property_name)` para extrair o valor property_name; ou então, usa o método `put(property_name, property_value)` para

definir a propriedade como o valor desejado. Ao definir propriedades, o nome da propriedade ou o valor da propriedade não pode ser nulo; se um deles for nulo, uma Exceção com uma mensagem apropriada será lançada.

O conjunto de propriedades disponível é o seguinte:

Tabela 6. Propriedades do Componente Disponíveis Durante a Execução do AssemblyLine

Propriedade	Uso
numErrors	O número de erros ocorridos.
numAdd	O número total de entradas que o AssemblyLine incluiu (executado por Conectores no modo AddOnly).
numModify	O número total de entradas que o AssemblyLine modificou (executado por Conectores no modo Atualizar).
numDelete	O número total de entradas que o AssemblyLine excluiu (executado por Conectores no modo Excluir).
numGet	O número total de entradas que o AssemblyLine recuperou (executado por Conectores no modo Repetidor).
numGetTries	O número total de vezes que o AssemblyLine tentou recuperar uma entrada (executado por Conectores no modo Repetidor).
numGetClient	O número total de clientes aceitos (disponível para Conectores no modo Servidor).
numGetClientTries	O número total de vezes que o AssemblyLine tentou obter o próximo cliente conectado (executado por Conectores no modo Servidor).
numCallreply	O número total de operações Chamar/Responder que o AssemblyLine executou (executado por Conectores no modo CallReply).
numLookup	O número total de operações de Consulta que o AssemblyLine executou (executado por Conectores no modo Atualizar/Excluir/Consulta).
numNoChange	O número total de entradas que o AssemblyLine processou, mas deixou inalterado.
numSkipped	O número total de entradas que o AssemblyLine ignorou.
numIgnored	O número total de entradas que o AssemblyLine ignorou (executado por Conectores no modo Atualizar/Delta).
lastCallStatus	Contém o status da execução do AL. Não é apenas uma propriedade somente leitura e pode ser modificada por você. O valor dessa propriedade é “fail” ou “success”, dependendo da execução do AL.
lastConn	A entrada Conn a partir da última operação do Conector. Antes da primeira operação do Conector, lastConn tem um valor igual a nulo.
lastError	O último erro como um objeto Java.
hooksInvoked	Uma java.util.List dos nomes dos ganchos chamados na última vez em que o Componente foi chamado. Os nomes são nomes internos.
success	Essa propriedade é definida como true se a última operação foi bem-sucedida e false se for malsucedida.
endOfData	True quando o Componente Repetidor atingiu o Fim dos Dados e falso se não atingiu. A alteração dessa propriedade não tem efeito.

Nota: Se for feita uma tentativa de alterar uma propriedade somente leitura, então uma Exceção com a mensagem apropriada será lançada.

Exemplo

Para ilustrar o uso dessas Propriedades do Componente, vamos supor que você tenha um Conector do Sistema de Arquivos chamado *FS* e alguns Componentes do Script. O código JavaScript a seguir está no gancho "GetNext Successful" do FS:

```
if(work.getString("ID") == null)
throw new java.lang.Exception("Missing ID");
//for the AL Cycle to execute properly I need an ID, so throw an Exception
```

E esse código JavaScript está no gancho "DefaultOnError" do FS:

```
if(FS.get("lastError").getMessage().equals("Missing ID")) {
//I could fix this by adding an ID which would help AL execution
 work.setAttribute("ID", "SomeID"); //add the ID
 if(FS.get("fixErrors") == null) {
  var vector = new java.util.Vector();
  vector.add(FS.get("lastError"));
  FS.put("fixErrors", vector);//save all fixed errors in my custom property
 } else {//I have previously fixed similar error
  var vector = FS.get("fixErrors");
  vector.add(FS.get("lastError"));
  FS.put("fixErrors", vector); //save all fixed errors in my custom property
 }
 FS.put("lastCallStatus", "success");
} else {//I could not fix this error
 if(FS.get("notFixErrors") == null) {
```

```
  var vector = new java.util.Vector();
  vector.add(FS.get("lastError"));
  FS.put("notFixErrors", vector); //save all not fixed errors in my custom property
 } else {
  var vector = FS.get("notFixErrors");
  vector.add(FS.get("lastError"));
  FS.put("notFixErrors", vector); //save all not fixed errors in my custom property
 }
 FS.put("lastCallStatus", "fail");
}
```

Finalmente, em um Componente do Script, considere o seguinte código:

```
main.logmsg("AL Cycle status: " + FS.get("lastCallStatus"));
//print the AL status for this AL Cycle
//I can also report all errors which have occurred
// during the AL Execution through my custom property, "vector"
```

Pontos de Controle para o Script

Fazendo um Script em um AssemblyLine

Os AssemblyLines fornecem um comportamento padrão pré-programado. Se você desejar desviar desse comportamento padrão, poderá fazer isso implementando sua própria lógica de negócios por meio de script.

Componente do Script

Você pode incluir Componentes de Script em seu AssemblyLine, além de Conectores, clicando com o botão direito do mouse em **Inserir Componentes...** > **Scripts** > **Script** no Componente ou seção apropriado, na janela Componentes do AssemblyLine, no Editor de AssemblyLine. O Componente de Script é iniciado uma vez para cada entrada processada pelo AssemblyLine e pode ser colocado em qualquer lugar no AssemblyLine.

Nota: Os Repetidores ainda são processados primeiro, mesmo se você colocar seu Componente de Script antes deles no AssemblyLine.

Para obter informações adicionais, consulte “Componentes de Script” na página 23.

Ganchos do AssemblyLine

Os Ganchos do AssemblyLine (ou seja, Ganchos que se aplicam ao AssemblyLine como um todo, não a qualquer Componente individual) são encontrados na guia **Ganchos** do AssemblyLine. Esses Ganchos são todos executados apenas uma vez por execução do AssemblyLine ou, no caso de **Solicitação de Encerramento**, sempre que o AssemblyLine for solicitada a encerrar por algum processo externo. Entretanto, se você iniciar sua AssemblyLine várias vezes (por exemplo, usando o Conector do AssemblyLine), iniciará então os Ganchos várias vezes também.

Os Ganchos dentro de um Conector só são avaliados e executados (onde definidos e não vazios) quando o Conector, no qual eles estão definidos, estiver em execução. Consulte “Fazendo Script em um Conector” na página 69 para obter informações adicionais.

Ganchos de Servidor

Os Ganchos de Servidor permitem gravar código JavaScript para responder a eventos e erros que ocorrem no nível do servidor. Diferentes dos Ganchos de AssemblyLine e de Componente, os Ganchos de Servidor são armazenados em arquivos de script separados. Esses arquivos são mantidos na pasta *serverhooks* no diretório de solução atual e devem conter funções de script nomeadas especificamente. As instâncias de servidor e de configuração do IBM Security Directory Integrator fornecem um método para os componentes do IBM Security Directory Integrator chamarem Ganchos customizados no nível do Servidor. Um Gancho de Servidor é um nome de função definido em um arquivo de script. As

implementações de função são fornecidas simplesmente soltando os arquivos de script no diretório "serverhooks" do diretório de solução.

Além destes Ganchos serem chamados pelo Servidor quando ocorrem eventos específicos, eles também podem ser chamados a partir de seus scripts. As chamadas a esses Ganchos são sincronizadas para evitar potenciais problemas de multiencadeamento.

Na inicialização, o IBM Security Directory Integrator carrega e executa todos os scripts do usuário no subdiretório *serverhooks*. Os scripts podem conter, ou não, declarações de função. Um script que não possui declarações de função é executado uma vez na inicialização, antes de quaisquer instâncias de configuração serem iniciadas. O código que define funções padrão de Gancho do Servidor IBM Security Directory Integrator é prefixado com "SDI_". e é executado em vários pontos durante a operação.

Todas as funções de Gancho do Servidor IBM Security Directory Integrator possuem a seguinte assinatura do JavaScript:

```
/**
   * @param NameOfFunction The configuration instance invoking the function
  * @param source The component invoking the function
  * @param user Arbitrary parameter information from the source
  */
 function SDI_functionName(Name_of_function, source, user) {
 }
```

Os parâmetros "NameOfFunction" e "source" sempre fornecem acesso à Instância de Configuração e ao componente de chamada, respectivamente. O parâmetro "user" é utilizado para diferentes propósitos nas várias funções de Gancho.

Os nomes de função padrão a seguir são invocados por vários componentes do IBM Security Directory Integrator:

Nome da Função	Chamado por (Origem)	Parâmetro do Usuário e Valor Esperado
SDI_ALStarted	Instância de Configuração	Chamado quando um AssemblyLine é iniciado. user = O AssemblyLine que foi iniciado *valor de retorno ignorado*
SDI_ALStopped	Instância de Configuração	Chamado quando um AssemblyLine é parado. user = O AssemblyLine que foi parado *valor de retorno ignorado*
SDI_ConfigStarted	Servidor	Chamado quando uma instância de Configuração é iniciada. user = A instância de configuração *valor de retorno ignorado*
SDI_ConfigStopped	Servidor	Chamado depois que uma Configuração instanciada é parada. user = A instância de configuração *valor de retorno ignorado*
SDI_Shutdown	Servidor/Instância de Configuração	Chamado logo antes de o servidor IBM Security Directory Integrator finalizar a JVM (por exemplo, System.exit()). user = Status de saída (inteiro) *valor de retorno ignorado*

O acesso às funções de Gancho do Servidor IBM Security Directory Integrator é fornecido por meio do método `main.invokeServerHook()`. Essa função é sincronizada para impedir que mais de um encadeamento execute um Gancho ao mesmo tempo. Todas as chamadas são solicitadas sincronizadamente para que o responsável pela chamada aguarde o retorno da função. Como resultado, deve-se tomar cuidado para não gastar muito tempo em um Gancho de servidor.

Conforme mencionado anteriormente, scripts são definidos e disponibilizados através da criação de arquivos no subdiretório "serverhooks" do diretório de soluções. Os scripts que contêm informações sigilosas devem ser criptografados com a API do Servidor antes de serem incluídos no diretório. A ferramenta `serverapi/cryptoutils` está disponível para criptografar arquivos de script. Observe que o IBM Security Directory Integrator tenta decriptografar automaticamente os arquivos com a extensão .jse, portanto, os arquivos criptografados devem ter de preferência essa extensão.

Além disso, os arquivos no diretório `serverhooks` são carregados e executados depois que seus nomes são classificados utilizando a distinção entre maiúsculas e minúsculas com a sequência de intercalação padrão para a plataforma. Todos os arquivos no diretório de nível superior são carregados antes do processamento dos arquivos em quaisquer subdiretórios.

Alguns exemplos de Gancho de Servidor são:

- Um objeto customizado que você sempre deseja que seja carregado no IBM Security Directory Integrator para uso em seu próprio script poderia ser instanciado a partir de um fragmento de JavaScript em gancho para o Gancho de servidor na inicialização do IBM Security Directory Integrator. Isso fornece mais controle do que simplesmente faz referência à classe sob a pasta de Bibliotecas Java no Navegador de Configuração.
- Um ou mais ALs customizados que você inicia criam um log de auditoria para esses eventos ou propagam esses eventos para outros sistemas utilizando algum transporte (SNMP, HTTP, JMS e outros).
- Uma política de segurança corporativa implementada que é chamada toda vez que uma Configuração é carregada ou o AL é iniciado.

Chamando Ganchos de Servidor a partir de um Script: A classe `com.ibm.di.server.RS` (variável de script "main") possui um método para chamar Ganchos de Servidor:

```
/**
 * Invokes a server hook.
 *
 * @param name The name of the hook (also the filename)
 * @param caller The object invoking the hook
 * @param userInfo Arbitrary information to the hook from the caller
 */

public Object invokeServerHook(
 String name,
 Object caller,
 Object userInfo) throws Exception;
```

Essa chamada pode retornar um Objeto Java (qualquer tipo), portanto, mesmo que o IBM Security Directory Integrator ignore isso durante a execução do Gancho de Servidor, é possível usar os valores retornados em suas próprias chamadas de script.

Em um script em um AssemblyLine, você pode fazer isto:

```
main.invokeServerHook("MyCustomHook", this, "custom information");
```

Acessando Componentes do AL dentro do AssemblyLine

Cada componente do AL está disponível como uma variável pré-registrada do script, com o nome escolhido para o componente.

Observe que você pode carregar dinamicamente os componentes com chamadas em script para funções como `system.getConnector()`, embora isso não seja para usuários experientes.[8]

Transmissão dos Parâmetros do AssemblyLine

Há três maneiras de obter dados em um AssemblyLine:

- Gerando sua própria entrada inicial dentro do AssemblyLine, por exemplo, em um script Prólogo.
- Alimentados de um ou mais Repetidores.
- Iniciando o AssemblyLine com parâmetros de um outro AssemblyLine, usando o Conector AL ou o Componente de Função AL ou usando uma chamada de API.

Se você desejar iniciar um AssemblyLine com parâmetros de outro AssemblyLine, então terá duas opções:

- Usar o TCB (**Task Call Block**), que é o método mais indicado. O TCB é detalhado a seguir.
- Fornecer uma Entrada de Trabalho Inicial diretamente.

  Nota: Essas duas opções são fornecidas para compatibilidade com versões anteriores.

TCB (Task Call Block)

O Task Call Block (TCB) é um tipo especial de objeto de Entrada usado por um responsável pela chamada para configurar vários parâmetros para um AssemblyLine.

Usação Básica: O Task Call Block (TCB) é um tipo especial de objeto de Entrada usado por um responsável pela chamada para configurar vários parâmetros para um AssemblyLine. O TCB pode fornecer uma lista de parâmetros de entrada ou saída especificados por um AssemblyLine, incluindo códigos de operação definidos na guia **Operações** do AssemblyLine, além de possibilitar que o responsável pela chamada configure parâmetros para os Conectores do AssemblyLine. O TCB é formado pelas seguintes seções lógicas:

- A Entrada de Trabalho Inicial transmitida ao AssemblyLine: `tcb.setInitalWorkEntry()`
- Os parâmetros do Conector: `tcb.setConnectorParameter()`
- As regras de mapeamento de entrada ou saída para o AssemblyLine, que são configuradas no Config Editor sob a guia **Operações**
- Um objeto opcional fornecido pelo usuário *accumulator* que recebe todas as entradas de trabalho do AssemblyLine: `tcb.setAccumulator()`

8. O objeto Conector obtido dessa chamada é um objeto da *Interface de Conector* e é a parte específica de um Conector do AssemblyLine. Quando você altera o *tipo* de qualquer Conector, está realmente trocando sua inteligência de origem de dados (a Interface de Conector) que fornece a funcionalidade para acessar dados em um sistema, serviço ou armazém de dados específico. A maior parte da funcionalidade de um Conector do AssemblyLine, incluindo os mapas de atributos, os Critérios de Link e os Ganchos, é fornecida pelo kernel IBM Security Directory Integrator e é mantida intacta quando você alternar os tipos de Conector.

Por exemplo, o início de um AssemblyLine com uma Entrada de Trabalho Inicial e a configuração do parâmetro *filePath* de um Conector chamado MyInput para d:\myinput.txt são feitos com o seguinte código:

```
var tcb = system.newTCB();                      // Create a new TCB
var myIWE = system.newEntry();                  // Create a new entry object
myIWE.setAttribute("name","John Doe");          // Add an attribute to myIWE
tcb.setInitialWorkEntry ( myIWE );              // Set the IWE and parameters
// Note that since this is a JavaScript string, we must "escape" the forward slash
//      or use a backslash (Windows syntax)
tcb.setConnectorParameter ( "MyInput", "filePath", "d:\\myinput.txt" );

var al = main.startAL ( "MyAssemblyLine", tcb ); // Start the AL with the tcb
al.join();                                      // Wait for AL to finish
```

Iniciando um AssemblyLine com Operações: Os AssemblyLines podem ser definidos com *Operações*; um conceito através do qual vários Mapas de Entrada são definidos para o AssemblyLine. Dependendo de como o AssemblyLine é chamado, um Mapa de Entrada diferente é ativado. No AssemblyLine, você precisará verificar a op-entry para descobrir qual operação está ativa e usar os Componentes de Ramificação para padronizar o fluxo no AssemblyLine para a operação relevante.

Uma maneira de iniciar um AssemblyLine com uma Operação é por meio de um TCB e de um código de script.

Se um AssemblyLine nomeado "al1" tiver as seguintes operações: "Padrão", "Op1" e "Op2", esse script iniciará o AL com a operação configurada como "Op1":

```
var tcb = system.newTCB("al1");
tcb.setALOperation("Op1");
main.startAL(tcb);
```

Não especificar qualquer operação iniciará o AL com a operação configurada como "Padrão":

```
var tcb = system.newTCB("al1");
main.startAL(tcb);
```

No caso em que o AL não possui uma operação Padrão (por exemplo, somente as operações "Op1" e "Op2"), o segundo script lançará uma exceção.

Informações adicionais sobre o conceito de Operações do AssemblyLine estão disponíveis na seção intitulada "Creating new components using Adapters" no *Referência*.

Usando um Acumulador: Como observado anteriormente, é possível também transmitir um objeto acumulador para um AssemblyLine com o TCB. Um acumulador pode ser qualquer uma das seguintes classes ou interfaces:

java.util.Collection
Todas as entradas de trabalho são clonadas e incluídas na coleta; por exemplo, ArrayList, Vector e outras.

com.ibm.di.connector.ConnectorInterface (Interface de Conector)
O método putEntry() para essa Interface de Conector é chamado com a entrada de Trabalho no final de cada ciclo do AssemblyLine.

com.ibm.di.parser.ParserInterface (Analisador)
O método writeEntry() é chamado para esse Analisador com a entrada de Trabalho no final de cada ciclo do AssemblyLine.

com.ibm.di.server.AssemblyLine Component (Conector do AssemblyLine)
O método add() é chamado para esse Conector do AssemblyLine com a entrada de Trabalho no final de cada ciclo do AssemblyLine.

Se o acumulador não for uma dessas classes ou interfaces, uma exceção será retornada.

Por exemplo, para acumular todas as entradas de trabalho de um AssemblyLine em um arquivo XML, você pode usar o seguinte script:

```
var parser = system.getParser ( "example_name.XML" ); // Get a Parser
// Set it up to write to file
parser.setOutputStream ( new java.io.FileOutputStream ( "d:/accum.xml" ));
parser.initParser();                                  // Initialize it.
tcb.setAccumulator ( parser );                        // Set Parser to tcb

var al = main.startAL ( "MyAssemblyLine", tcb );      // Start AL with tcb
al.join();                                            // Wait for AL to finish

parser.closeParser();    // Close the parser - this flushes and
             // closes the output file
```

Além disso, você pode configurar um Conector em vez de programar o Analisador manualmente, como no script a seguir:

```
var connector = system.getConnector("myFileSysConnWithXMLParser");
tcb.setAccumulator ( connector );

var al = main.startAL( "MyAssemblyLine", tcb);
al.join();

connector.terminate();
```

Geralmente, você inicializa o TCB, que é então usado pelo AssemblyLine. Se o AssemblyLine tiver uma especificação de Operações, o TCB remapeará os atributos de entrada para a Entrada de Trabalho Inicial conforme esperado pelo AssemblyLine e, da mesma forma, para configurar o objeto de resultado. Isso é feito para que a interface de chamada externa para um an AssemblyLine possa permanecer igual, mesmo que os nomes de entrada de Trabalho internos sejam alterados no AssemblyLine. Assim que o TCB é transmitido a um AssemblyLine, você não deve esperar mais nada do TCB. Use `getResult()` e `getStats()` do AssemblyLine para recuperar as estatísticas e o objeto de resultado.

O mapeamento de resultado do TCB é executado antes do Epilog para que você ainda possa acessar o resultado final antes do responsável pela chamada do AssemblyLine.

Desativando Componentes do AssemblyLine: No IBM Security Directory Integrator, você pode especificar que certos componentes do AssemblyLine não devem ser criados ou inicializados na inicialização do AssemblyLine. Isso é feito desativando esses componentes usando o TCB.

Os componentes do AssemblyLine são ativados por padrão.

Para ativar ou desativar um componente no AssemblyLine, você deve chamar o método `com.ibm.di.server.TaskCallBlock.setComponentEnabled(String name, boolean enabled)` no objeto TCB do AssemblyLine. O argumento name do método especifica o nome do componente a ser ativado ou desativado. O argumento ativado do método especifica se o componente será ativado ou desativado.

A ativação ou desativação real dos componentes do AssemblyLine acontece no método `com.ibm.di.server.TaskCallBlock.applyALSettings(AssemblyLineConfig alc)`. Esse método é chamado na inicialização do AssemblyLine. À medida que a inicialização do AssemblyLine progride, os componentes que foram marcados como Desativados não são criados/inicializados.

Se um componente LOOP for desativado, todos os componentes contidos nesse LOOP também serão desativados.

Mesmo se um componente for desativado do Editor de Configuração, ele pode ser ativado usando essa chamada de script:

```
com.ibm.di.server.TaskCallBlock.setComponentEnabled(String name, boolean enabled)
```

Fornecendo uma IWE (Entrada de Trabalho Inicial)

Fornecer uma Entrada de Trabalho Inicial (IWE) é uma maneira alternativa de transmitir parâmetros usando um TCB e tem suporte quanto à compatibilidade com versões anteriores.

Quando um AssemblyLine for iniciado com a chamada **system.startAL()** a partir de um script, o AssemblyLine pode ainda ter os parâmetros transmitidos configurando valores de atributos ou de propriedades na Entrada de Trabalho Inicial, que é acessada por meio da variável *work*. Ela é então a tarefa a ser aplicada nesses valores para definir os parâmetros do Conector; por exemplo, no Gancho **Prólogo – Inic** usando a função *connectorName*.`setParam()`.

Nota: Você deve limpar a Entrada de trabalho com a chamada `task.setWork(null)`; caso contrário, os Repetidores do AssemblyLine serão transmitidos no primeiro ciclo.

Você pode examinar o resultado do AssemblyLine, que é a Entrada de trabalho quando o AssemblyLine para, usando a função `getResult()`. Consulte também "Conector fornecido pelo tempo de execução" em *Referência*.

Segue um exemplo de transmissão em um valor de parâmetro do Conector com uma IWE:

```
var entry = system.newEntry();
entry.setAttribute ("userNameForLookup", "John Doe");

// Here we start the AssemblyLine
var al = main.startAL ( "EmailLookupAL", entry );

// wait for al to finish
al.join();

var result = al.getResult();

// assume al sets the mail attribute in its working entry
task.logmsg ("Returned email = " + result.getString("mail"));
```

Fazendo Script em um Conector

Mapa de Entrada e Mapa de Saída

O mapeamento do atributo customizado é executado nestas guias. Quando o atributo é selecionado, você deve selecionar a caixa de opções **Mapeamento Avançado (Advanced Mapping)** e inserir seu script na janela de edição. Lembre-se de que antes de fazer todo o processamento necessário, você deve atribuir o valor do resultado obtido ao `ret.value`, por exemplo:

```
   ...
ret.value = myResultValue;
```

Alternativamente, no IBM Security Directory Integrator, é possível usar a palavra-chave *return* seguida pelo valor simples que você gostaria de passar como resultado:

```
return "mystring";
```

Ganchos de Conector

Os Ganchos dão a você o meio de responder a determinados eventos que ocorrem e substituir a funcionalidade básica de um Conector. Você tem acesso aos objetos globais ao gerar o script de Ganchos, embora alguns dos objetos padrão podem não estar disponíveis em todos os Ganchos. Para obter detalhes sobre a disponibilidade de objetos temporários, consulte "Fluxogramas do AssemblyLine e dos Modos de Conectores", na publicação *Referência*. Você também tem controle integrado sobre o ambiente, o AssemblyLine, o Conector, as entradas e os atributos. Os Ganchos dão a você uma diversidade de pontos de controle para personalizar o fluxo do processo. Consulte o "Fluxo do AssemblyLine e Ganchos" na página 33.

Definindo Parâmetros Internos por Script

É possível configurar os parâmetros de conexão para um Conector utilizando o seguinte script:

```
myConnector.setParam ( "filePath", "examples/scripting/sample.csv" );
```

Isso é geralmente algo que você executa no Prólogo, mas pode ser muito útil também enquanto o AssemblyLine está em execução, desde que o Conector seja parado e reinicializado.

```
myConnector.terminate();
myConnector.setParam ( "filePath", "examples/scripting/sample.csv" );
myConnector.initialize(null);
```

Fazendo Script em um Analisador

Fazer script em um Analisador se refere, realmente, à implementação do seu próprio Analisador pelo script. Uma descrição desse processo está incluída na seção chamada "Analisador de Script" no *Referência*.

Java + Script ≠ JavaScript

JavaScript **não** é Java. Ele pode parecer como Java, mas, na verdade, essa semelhança é suficiente para causar confusão. Quando foi criado pela primeira vez pela Netscape, o JavaScript era originalmente chamado de *Live!Script*. Embora exista um amplo suporte para o JavaScript, você perceberá que existem *dialetos*; por exemplo, a versão da Microsoft, chamada de *JScript*. Existe uma definição padrão, conhecida como *ECMAScript* e você pode encontrar sua especificação nesta URL: http://www.ecma-international.org/

Embora a sintaxe seja semelhante, o Java e o JavaScript lidam com dados e tipos de dados de forma diferente. Essa é uma das principais origens de confusão e, com ela, erros ao trabalhar com JavaScript.

Representação de Dados

O Java suporta elementos chamados de *primitivos*, que consistem em valores simples, como inteiros assinados, valores decimais e bytes únicos. Os primitivos

são fornecem nenhuma funcionalidade, apenas um conteúdo de dados não-complexo. Você pode usá-los em cálculos e expressões, designá-los a variáveis e transmiti-los em chamadas de função. O Java também fornece uma grande variedade de objetos que não só transportam conteúdo de dados (até mesmo dados muito complexos), como também fornecem informações no formato de funções de objeto, conhecidas também como métodos.

Quando uma chamada de script é feita para `task.logmsg( "Hello, World" )`, o método `logmsg()` do objeto de tarefa está sendo realmente chamado.

Muitos primitivos Java possuem objetos correspondentes. Um exemplo é um número inteiro, que pode ser usado em seu formato primitivo *(int)* ou pela manipulação de objetos `java.lang.Integer`.

JavaScript não usa o conceito de primitivos. Em vez disso, todos os dados são representados como objetos JavaScript. Além disso, o JavaScript possui apenas alguns objetos nativos, em contraste com o extenso vocabulário de dados do Java. Portanto, enquanto o Java diferencia objetos numéricos não fracionais e seus equivalentes decimais – fazendo até mesmo diferenciações entre tipos assinados e não assinados, além de oferecer objetos semelhantes para diferentes níveis de precisão –, o JavaScript consolida todos os valores numéricos em um único tipo de objeto, denominado *Number*.

Como resultado, você pode obter resultados aparentemente incorretos ao comparar valores numéricos em JavaScript:

```
if (myVal == 3) {
 // do something here if myVal equals 3
}
```

Se o elemento `myVal` tiver sido definido por uma operação aritmética ou se fizer referência a um objeto decimal Java, o valor do objeto poderá ser 3.00001 ou 2.99999. Embora esse resultado esteja muito próximo de 3, ele não será aprovado no teste de equivalência anterior. Para evitar esse problema específico, é possível converter o operando em um objeto Inteiro Java, para garantir um valor assinado e não fracional. Dessa forma, sua expressão Booleana apresentará o comportamento esperado.

```
if (java.lang.Integer( myVal ) == 3) { ...
```

Outra alternativa é garantir que as variáveis façam referência aos objetos Java apropriados com os quais começar. Em geral, convém conhecer todos os tipos de objetos que estão sendo usados.

Chamadas de Funções Ambíguas

O Java também fornece um tipo primitivo chamado de *char*, que pode conter um único valor do caractere. Uma coleção de caracteres poderia ser representada em Java como uma matriz de caracteres primitivos ou poderia ser manipulada como um objeto `java.lang.String`. Conforme mencionado anteriormente, o JavaScript não compreende primitivos. Os dados de caractere devem ser manipulados com o uso do objeto *cadeia* do JavaScript. Mesmo que você especifique um único caractere em JavaScript ("a"), ele será considerado uma *cadeia*.

Agora, considere que, quando uma função Java é chamada a partir de scripts, ela é correspondida ao método real com o uso do nome da função e também do número e dos tipos de parâmetros usados na chamada. Essa correspondência é realizada pela extensão LiveConnect do JavaScript. O LiveConnect faz o melhor possível

para descobrir qual assinatura de função está sendo referenciada, o que não é uma tarefa fácil, pois o Java e o JavaScript representam tipos de parâmetros de maneiras diferentes. Mas o JavaScript e o LiveConnect fazem algumas conversões internas para você, tentando corresponder tipos de dados Java e JavaScript.

Surge um problema quando você tem múltiplas versões de um único método, cada um obtendo um conjunto diferente de parâmetros; em especial, se duas funções tiverem a mesma quantidade de parâmetros, mas de tipos diferentes, **e** se esses tipos não forem diferenciados em JavaScript. Este é um exemplo de script que executará uma criptografia MD5 de uma cadeia.

```
// Create MessageDigest object for MD5 hash
var md = new java.security.MessageDigest.getInstance( "MD5" );

// Get the EID attribute value as byte array.
var ab = java.lang.String( "message to encrypt" ).getBytes();

md.update( ab );

var retHash = md.digest();
```

A chamada `update()` anterior apresentará uma falha, com uma exceção de avaliação indicando que a chamada de função é ambígua. Isso ocorre porque o objeto `MessageDigest` possui várias versões dessa função com assinaturas semelhantes: uma aceitando um byte único e outra esperando uma matriz de byte (`byte[]`). Você pode contornar isso se puder encontrar outra variante do mesmo método obtendo uma quantidade diferente de parâmetros, além de fornecer uma assinatura identificada com exclusividade. Felizmente, `MessageDigest` fornece alguma conveniência: uma versão de `update()` que obtém uma matriz de byte mais alguns valores numéricos (parâmetros de deslocamento e de comprimento). Portanto, você pode alterar o seu código para usar essa chamada em vez de:

```
md.update( ab, 0, ab.length );
```

Por último, sempre é possível especificar a assinatura exata do método Java que você deseja usar, delimitando entre aspas, dentro de colchetes, após o objeto:

```
md["update(byte[])"](ab);
```

Nesse caso, estamos chamando a versão da função `update()` declarada com um parâmetro de matriz de bytes únicos.

Dados de Caractere/Cadeia em Java versus Cadeias JavaScript

Tanto o Java quanto o JavaScript fornecem um objeto de Cadeia. Embora esses dois tipos de objetos de Cadeia apresentem um comportamento semelhante e ofereçam diversas funções idênticas, eles são diferentes em vários aspectos. Por exemplo, cada tipo de objeto fornece um mecanismo diferente para retornar o comprimento de uma cadeia. Com Cadeias Java, o método `length()` deve ser usado. As cadeias JavaScript por outro lado têm `comprimento` *variável*.

```
var jStr_1 = new java.lang.String( "Hello, World" ); // Java
String
task.logmsg( "the length of jStr_1 is " + jStr_1.length() );

var jsStr_A = "Hello, World"; // JavaScript String
task.logmsg( "the length of jsStr_A is " + jsStr_A.length );
```

Essa diferença sutil pode gerar erros confusos de sintaxe. Tentar chamar `jsStr_A.length()` gerará um erro de tempo de execução, já que esse objeto não possui um método `length()`.

Erros ainda mais confusos podem ocorrer com comparações de cadeias.

```
var jsStr_A = "Hello, World"; // JavaScript String
var jsStr_B = "Hello, World"; // JavaScript String

if ( jsStr_A == jsStr_B )
 task.logmsg( "TRUE" );
else
 task.logmsg( "FALSE" );
```

Conforme esperado, você obterá um resultado "TRUE" a partir do fragmento anterior. Entretanto, as coisas funcionam de maneira um pouco diferente com Cadeias Java.

```
var jStr_1 = java.lang.String( "Hello, World" ); // Java
String
var jStr_2 = java.lang.String( "Hello, World" ); // Java String

if ( jStr_1 == jStr_2 )
 task.logmsg( "TRUE" );
else
 task.logmsg( "FALSE" );
```

Isso resultará em "FALSE", pois o operador de equivalência anterior fará uma comparação para verificar se ambas as variáveis fazem referência ao *mesmo objeto* na memória, em vez de corresponder seus valores. Para comparar valores de Cadeias Java, é necessário usar o método de Cadeia apropriado:

```
if ( jStr_1.equals( jStr_2 ) ) ...
```

Entretanto, isso não é tudo. O próximo fragmento de código gerará um resultado "TRUE":

```
var jsStr_A = "Hello, World"; // JavaScript String
var jStr_1 = java.lang.String( "Hello, World" ); // Java
String

if ( jsStr_A == jStr_1 )
 task.logmsg( "TRUE" );
else
 task.logmsg( "FALSE" );
```

Como o JavaScript não consegue operar em um tipo desconhecido como um objeto de Cadeia Java, ele primeiro converterá `jStr_1` em uma Cadeia JavaScript equivalente para fazer a avaliação.

Em resumo, lembre-se dos tipos de objetos com os quais você está trabalhando. Lembre-se de que as funções do IBM Security Directory Integrator sempre retornam Objetos Java. A memorização desses fatores ajudará a minimizar erros no código de script.

Escopo e Nomenclatura de Variáveis

O JavaScript é uma linguagem relativamente informal e não requer que variáveis sejam definidas antes de terem valores designados. Não força a verificação de tipo estrito nem sinaliza quando uma variável é redefinida. Isso torna o JavaScript rápido e fácil de trabalhar, mas pode levar rapidamente a um código ilegível e a uma confusão de erros, especialmente uma vez que você pode criar variáveis que substituem as internas.

Um erro que pode provocar depuração é quando você declara uma variável com o mesmo nome de uma interna, como `work`, `conn` e `current`, de modo que você precisará se familiarizar com os nomes reservados usados pelo IBM Security Directory Integrator.

Outro problema comum ocorre quando você cria novas variáveis que redefinem variáveis existentes, talvez usadas em Bibliotecas de Scripts ou Configs incluídos. Esses erros poderão ser evitados se você levar em consideração a nomenclatura de variáveis e seu *escopo*. O escopo define a esfera de influência de uma variável e no IBM Security Directory Integrator falamos sobre variáveis globais; aquelas que estão disponíveis em todos os Ganchos, Componentes do Script e mapas de atributos, e aquelas que são locais para uma função.

Para compreender melhor o escopo, é necessário compreender primeiro que cada AL possui o seu próprio Mecanismo de Script e, portanto, é executado em seu próprio contexto de script. Qualquer variável não definida especificamente como local dentro de uma declaração de função é global para esse Mecanismo de Script. Portanto, o código a seguir criará uma variável global:

```
myVar = "Know thyself";
```

Essa variável estará disponível desse ponto em diante durante o ciclo de vida do AL. Torná-la local requer duas etapas: usar a palavra-chave `var` ao declarar a variável e colocar a declaração dentro de uma função:

```
function myFunc() {
 var myVar = "Know thyself";
}
```

Agora, `myVar`, conforme definida anteriormente, deixará de existir após a inserção da chave de fechamento. Observe que colocar a variável dentro de uma função não é o suficiente. Também é necessário usar `var` para indicar que você está declarando uma nova variável local.

```
var glbVar = "This variable has global scope";
glbVar2 = "Another global variable";

function myFunc() {
    var lclVar = "Locally scoped within this block";
    glbVar3 = "This is global, since "var" was not used";
};
```

Desde que declare uma variável local dentro de uma função, você poderá denominá-la como desejar. Assim que ela sair do escopo, os tipos e os valores anteriores serão restaurados.

Mesmo que a palavra-chave `var` não seja necessária para a definição de variáveis globais, convém usá-la. Além disso, convém definir suas variáveis no início do script, incluindo comentários suficientes para que o leitor possa compreender a finalidade para a qual elas serão usadas. Essa abordagem não apenas aprimora a legibilidade de seu código, mas força também que você tome decisões conscientes sobre escopo e nomenclatura de variáveis.[9]

9. A nomenclatura de funções é ligeiramente diferente. Linguagens de programação, como o Java, identificam uma função com base na combinação do nome e do número e dos tipos de parâmetros. O JavaScript apenas usa o nome. Portanto, se houver várias definições da mesma função, o JavaScript apenas se "lembrará" da última — independentemente de essa definição ter ou não um número diferente de parâmetros.

Instanciando uma Classe Java

Você pode chamar classes Java externas; ou seja, aqueles que você incluiu no ambiente de tempo de execução do IBM Security Directory Integrator ou o CLASSPATH, usando o código de script.

Por exemplo, suponha que você queira usar o `java.io.FileReader` padrão, use o seguinte script:

```
var javafile = new java.io.FileReader ( "myfile" );
```

Você tem agora um objeto chamado *javafile*; usando-o, é possível chamar todos os métodos do objeto.

A mesma técnica é usada para instanciar seus próprios objetos:

```
var myfile = new my.FileReader("myfile");
```

Usando Valores Binários no Script

Os valores binários podem ser recuperados a partir de Atributos, usando a função `getObject()` da entrada. O próprio valor de Atributo binário é retornado como uma matriz de byte. Aqui está um exemplo em JavaScript:

```
var x = conn.getObject("objectGUID");
for ( i = 0; i < x.length; i++ )
{
   task.logmsg ("GUID[" + i + "]: " + x[i]);
}
```

Este exemplo grava alguns números variando entre -128 e 127 no arquivo de log. Você pode desejar fazer algo a mais com seus dados. Se você tiver lido uma senha a partir de um Conector que a armazenou como uma ByteArray, poderá convertê-la em uma cadeia com este código:

```
password = system.arrayToString(conn.getObject("userpassword"));
```

Usando Valores de Data no Script

Ao trabalhar com datas no IBM Security Directory Integrator, isso implica em usar instâncias de `java.util.Date`. Provido com qualquer uma das linguagens de script disponíveis, você pode implementar seu próprio mecanismo para manipulação de datas; entretanto, esta não é uma prática comum.

O mecanismo de script do IBM Security Directory Integrator fornece um mecanismo de análise de datas. O objeto *system* possui um método `parseDate(date, format)` acessível a qualquer momento.

Nota: Ao receber uma instância de `java.util.Date`, você poderá utilizar as bibliotecas Java padrão e as classes para estender seu processamento.

Aqui está um simples exemplo de JavaScript que trata de datas. Este código pode ser colocado e inicializado a partir de qualquer ponto de controle de script:

```
var string_date1 = "07.09.1978";
var date1 = system.parseDate(string_date1, "dd.MM.yyyy");

var string_date2 = "1977.02.01";
var date2 = system.parseDate(string_date2, "yyyy.dd.MM");

task.logmsg(date1 + " is before " + date2 + ": " +
  date1.before(date2));
```

O código de script analisa primeiro dois valores de data (em formatos diferentes) para `java.util.Date`. Depois, ele usa o método `java.util.Date.before()` padrão para determinar se a primeira instância de data vem antes da segunda. A saída deste script é, então, impressa no arquivo de log.

Usando Valores de Ponto Flutuante no Script

Os exemplos a seguir demonstram como os valores de ponto flutuante podem ser usados dentro do código de script que você cria. Todos estes exemplos são implementados em JavaScript. Enquanto os mesmos exemplos podem ser repetidos usando várias outras linguagens de script, a sintaxe pode ser diferente. O script simples a seguir atribui valores de ponto flutuante a duas variáveis para encontrar sua média. Este código pode ser inicializado a partir de qualquer ponto de controle de script. A saída do arquivo de log é `" r = 3.85 "`.

```
var a = 5.5;
var b = 2.2;
var r = (a + b) / 2;
task.logmsg("r = " + r);
```

O próximo exemplo aumenta este script simples. Considere que, em seu Conector de entrada, existem atributos de multivalores denominados "Marks" contendo valores de cadeia (java.lang.String) que representam valores de ponto flutuante; uma situação comum. Esse atributo é mapeado para um atributo em seu Conector de saída chamado "AverageMark", que contém o valor médio de todos os valores do atributo "Marks". O seguinte código é usado no Mapeamento Avançado do atributo "AverageMark":

```
// First return the values of the "Marks" attribute
var values = work.getAttribute("Marks").getValues();

// Zero out counter and sum variables
var sum = 0;
var count = 0;

// Loop through the values, counting and summing them
for (i=0; i<values.length; i++)
{
   // use the Double() function to convert value to number
   sum = sum + new Number(java.lang.Double(values[i]));
   count++;
}

// Se contagem > 0, computar a média
var average = (count > 0) ? (sum / count) : 0;

// Return the computed average
ret.value = average;
```

A chamada central neste exemplo é o `java.lang.Double(values[i])` usado para converter o valor atualmente indexado de "Marks" em um valor numérico que pode ser usado no cálculo médio.

Capítulo 3. O Configuration Editor

O IBM Security Directory Integrator Eclipse Configuration Editor (CE) é a ferramenta principal para desenvolver soluções do IBM Security Directory Integrator. Ele permite criar, manter, testar e depurar arquivos de Configuração; ele é construído na plataforma Eclipse para fornecer um ambiente de desenvolvimento abrangente e extensível.

Para entender os conceitos apresentados aqui mais facilmente, você deve estar familiarizado com os conceitos comuns do Eclipse, tais como Editores, Visualizações e outros pontos de extensão comuns do Eclipse.

Modelo de Projeto

Com a chegada do Configuration Editor (CE) baseado no Eclipse, o desenvolvimento de soluções no IBM Security Directory Integrator (IBM Security Directory Integrator) não se baseia, como nas versões pré-V7.0, no desenvolvimento de arquivos de configuração simples, mas em um modelo de Projeto e em *Áreas de trabalho*. Usando o Modelo de projeto e a Área de trabalho, é possível fazer o seu trabalho de desenvolvimento; quando você estiver pronto para implementar a solução, extraia um arquivo de configuração da Área de trabalho e envie-o a um Servidor IBM Security Directory Integrator compatível: o ambiente de tempo de execução. Isso proporciona vários benefícios, tais como:

- Limpeza de arquivos de configuração; os elementos de configuração desnecessários e não usados não serão exportados da Área de trabalho;
- Edição mais eficiente de configurações do IBM Security Directory Integrator;
- Maior reutilização de componentes comuns;
- A possibilidade de usar sistemas de gerenciamento de código de origem, por exemplo, CVS.

Área de Trabalho

Quando você inicia o CE, será solicitado a selecionar um diretório da área de trabalho. O diretório da área de trabalho é onde o CE armazena todos os seus projetos e arquivos do IBM Security Directory Integrator. Ele pode ser diferente do diretório de solução, embora o diretório da área de trabalho possa estar muito bem contido dentro do diretório de solução. Na área de trabalho, o CE reúne arquivos para formar um arquivo de configuração de tempo de execução (`rs.xml`) que pode ser executado em um servidor IBM Security Directory Integrator. Os arquivos e projetos da área de trabalho podem ser considerados como a origem para vários arquivos de configuração de tempo de execução.

Diretório de Instalação, Diretório de Solução e o Diretório Ativo

Alguma confusão pode surgir do fato de haver três diretórios diferentes em execução. Conforme mencionado na seção anterior, o diretório da área de trabalho pertence ao CE em que os arquivos de projeto estão armazenados. Os diretórios de instalação e de solução pertencem ao servidor IBM Security Directory Integrator quando ele é executado. No entanto, quando você configura um conector no CE, por exemplo, geralmente usa a função de atributos de descoberta; isso é devido ao CE abrir o conector e executar operações no conector. Como alguns conectores usam nomes de arquivos para múltiplas finalidades, é possível fazer confusão

quando caminhos relativos são usados. Por essa razão, o diretório de trabalho padrão do CE deve sempre ser o mesmo que o diretório de soluções com o qual você está trabalhando inicialmente[10]. Principalmente porque é possível criar vários servidores e diretórios de soluções dentro do CE. Você pode ainda trabalhar com eles e executar AssemblyLines nesses servidores, mas os caminhos relativos em configurações não são mais iguais quando você usa conectores no CE, ao contrário de usá-los ao executar o AssemblyLine.

Considere uma configuração com um conector de arquivos usando `abc.txt` como seu arquivo de entrada. Quando você usa atributos de descoberta no CE, ele consultará um arquivo do diretório de trabalho do CE (o diretório de soluções de preferência especificado no momento da instalação). Quando você executa o AssemblyLine com esse conector, tudo parece bem. Em seguida, você cria um novo servidor com um diretório de soluções apontando para qualquer outro lugar em vez do diretório de trabalho atual. Você configura o conector e ele resolve o nome do arquivo para o local correto (ou seja, o arquivo abc reside em seu diretório de trabalho atual). No entanto, quando você executa o AssemblyLine, ele falha, pois não consegue encontrar o arquivo. Isso é devido ao novo servidor executar em um diretório de trabalho diferente (o diretório de soluções) que o CE.

Por padrão, o diretório de trabalho do CE é definido como o diretório de soluções do servidor padrão que é usado para executar AssemblyLines. Portanto, se você não alterar nada, não verá nenhum desses problemas, exceto quando cria novos diretórios de soluções ou altera o diretório de soluções do servidor padrão.

Ambiente de Trabalho

O ambiente de trabalho é o aplicativo principal de interface com o usuário no qual você executa toda a configuração e o desenvolvimento de soluções do IBM Security Directory Integrator. O ambiente de trabalho consiste em várias visualizações e editores que permitem fazer isso.

A Visualização Servidores do IBM Security Directory Integrator

Uma instância do servidor IBM Security Directory Integrator não é iniciada toda vez que você executa um AssemblyLine, tal como ocorria com as versões anteriores (pré-7.0). Em vez disso, um servidor IBM Security Directory Integrator é iniciado automaticamente quando você inicia o CE, que é usado para testar e executar AssemblyLines desenvolvidos. Ao contrário de versões anteriores, quando o objeto de suas execuções de teste é concluído, esse servidor não termina, mas permanece aguardando sua próxima execução de teste.

Na Visualização Servidores, você pode gerenciar definições do servidor. Os servidores definidos aqui podem ser servidores locais ou remotos e são usados pelos vários projetos IBM Security Directory Integrator criados. Os servidores que possuem um caminho de instalação definido são considerados servidores “locais” que podem ser ativados pelo CE.

Um servidor é definido automaticamente e nomeado como *Default*. Esse é o servidor padrão usado pelos novos projetos IBM Security Directory Integrator. O servidor padrão é criado com valores do arquivo de propriedades globais, usando

10. O diretório de trabalho não deve ser confundido com a área de trabalho. A área de trabalho é a área em que os arquivos do projeto são armazenados; o diretório de trabalho é a posição no sistema de arquivos que define todos os nomes de arquivos não totalmente qualificados. Por razões de portabilidade, esse deve ser o diretório de soluções.

o diretório de solução, conforme definido pelo usuário (por exemplo, da variável SDI_SOLDIR, que por sua vez é configurada pelo Instalador).

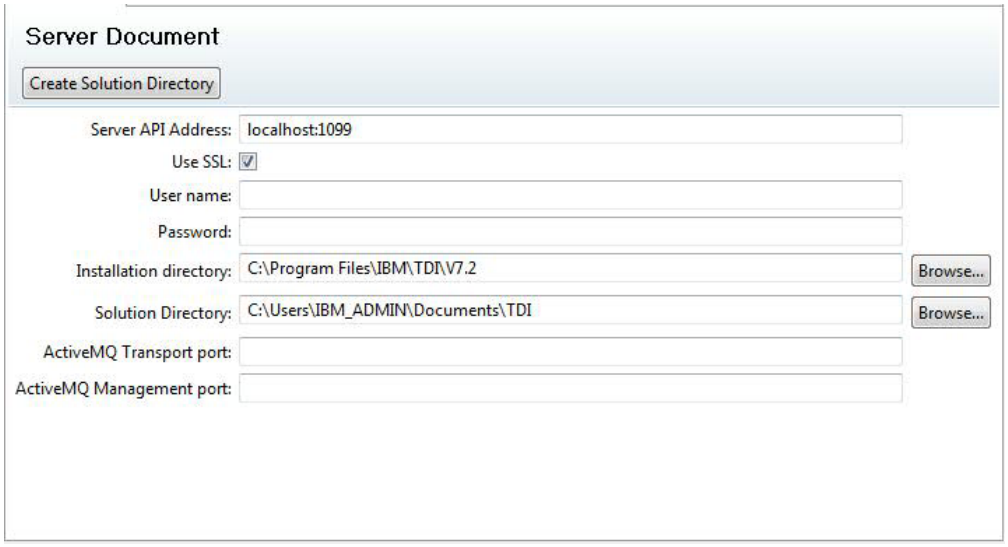


Figura 5. A Definição de Servidor Padrão do IBM Security Directory Integrator

O Projeto do IBM Security Directory Integrator

Para desenvolver uma solução do IBM Security Directory Integrator, você deve primeiramente criar um Projeto IBM Security Directory Integrator. Um projeto no Eclipse é uma coleta de arquivos e recursos relacionados.

Quando o projeto é criado, ele é preenchido com algumas pastas nas quais os objetos de configuração comuns estão localizados. O layout de um novo projeto IBM Security Directory Integrator é mostrado abaixo.

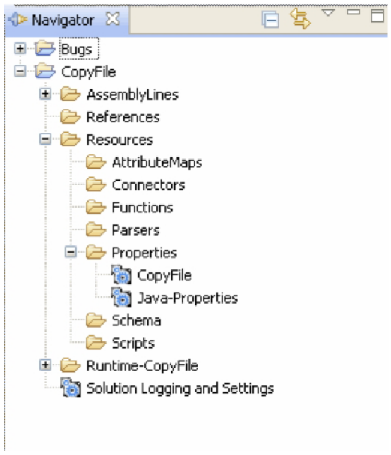


Figura 6. A Árvore do Projeto do IBM Security Directory Integrator

A pasta AssemblyLines contém os AssemblyLines do projeto; enquanto a pasta de recursos contém todos os componentes que são compartilhados ou usados pelos AssemblyLines ou que se aplicam à solução em geral: propriedades, parâmetros de criação de log e outros. Para criar novos recursos, use o assistente **Arquivo/Novo...**

no menu principal ou use os menus de contexto em cada pasta (ou seja, clique com o botão direito do mouse na pasta AssemblyLines para criar um novo AssemblyLine).

O diretório de Tempo de Execução (Tempo de Execução-*ProjectName*) é um diretório especial que contém o arquivo de configuração do runtime, assim como os arquivos de propriedades customizadas. Sempre que um componente é alterado no projeto (conector, AssemblyLine, arquivo de propriedades e outros) os arquivos relevantes nesse diretório também são atualizados. Todos os arquivos gerados são identificados por tags como arquivos derivados, o que significa que você receberá um aviso se tentar modificar o conteúdo. Saiba que se você alterar qualquer um desses arquivos, ele será sobrescrito. A intenção com o diretório de tempo de execução é criar um diretório de arquivos de tempo de execução para o projeto que pode ser copiado, ou retirado se você usar o controle de fonte, e usado por um servidor IBM Security Directory Integrator.

Arquivos de Configuração

O arquivo de configuração (Config) executado por um servidor é um documento XML composto no qual os AssemblyLines, conectores e outros fazem parte do mesmo documento. No IBM Security Directory Integrator Eclipse CE, cada um desses componentes é alocado em seu próprio arquivo físico. Esses arquivos contêm somente um objeto de configuração.

Uma das razões para dividir o arquivo de configuração em arquivos separados durante o desenvolvimento de solução é tornar o compartilhamento dos componentes mais fácil. Além disso, ter cada componente em seu próprio arquivo é perfeitamente apropriado para os sistemas de controle de origem, como o CVS, e também para o desenvolvimento multiusuário, em que diferentes pessoas trabalham em diferentes partes da solução ao mesmo tempo.

As configurações anteriores a esta versão podem ser importadas usando um assistente de importação. A configuração importada é dividida em arquivos de configuração individuais como resultado deste processo. Uma outra maneira é usar a opção **Arquivo** > **Abrir o Arquivo de Configuração do Security Directory Integrator**, que importará a configuração pré-7.0 para um novo projeto. Saiba, entretanto, que essa opção também configura a atualização automática novamente no arquivo de origem. Consulte a próxima seção para obter informações adicionais sobre o recurso de arquivo vinculado.

Arquivo de Configuração do Runtime

O arquivo de configuração do runtime fica oculto da visualização normal. Esse é o arquivo que os servidores IBM Security Directory Integrator usam para executar a solução e também é o arquivo em que versões anteriores (pré-7.0) trabalhariam diretamente. Sempre que você salvar um arquivo de configuração no CE, isso causará também uma atualização no arquivo de configuração do runtime. Esse arquivo também é transferido para um servidor IBM Security Directory Integrator para execução. Esse arquivo é mantido pelo Construtor de Projetos e não deve ser modificado pelo usuário final.

Você pode configurar seu projeto para exportar o arquivo de configuração do runtime automaticamente quando ele for alterado. Use o painel de propriedades do projeto para configurar o arquivo ao qual o projeto está vinculado.

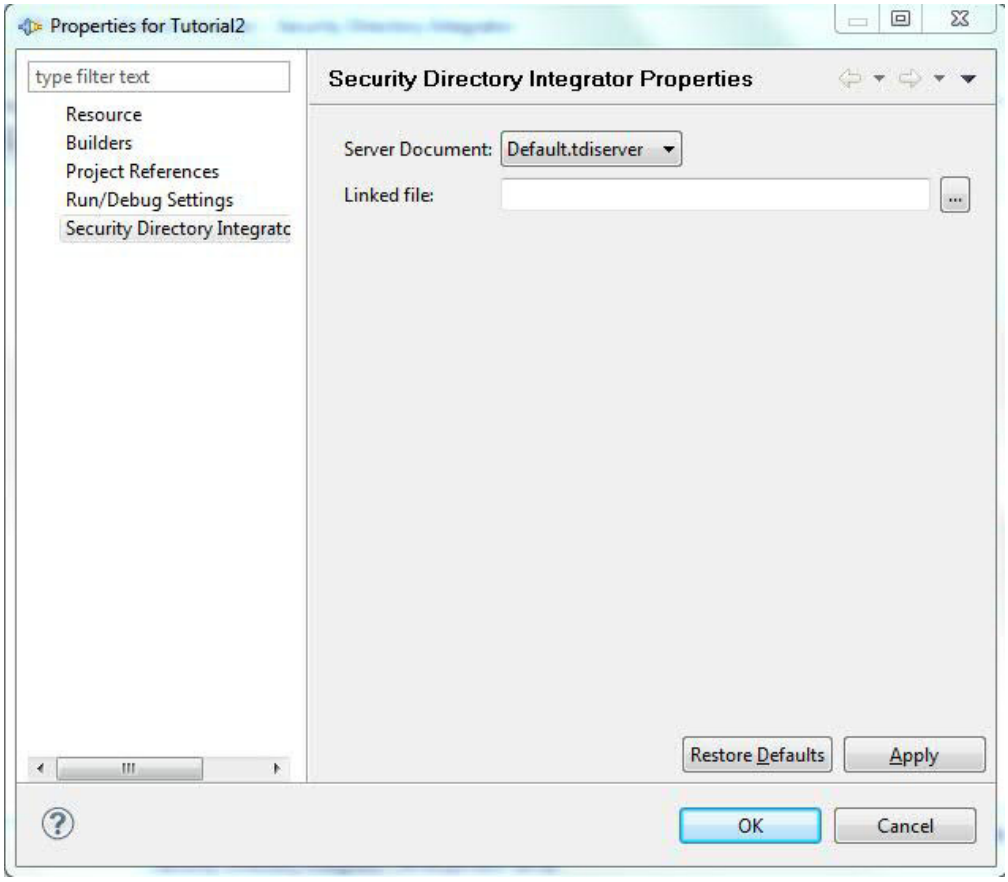


Figura 7. Janela Propriedades do Projeto do IBM Security Directory Integrator

Sempre que a configuração de tempo de execução é atualizada, ela também é copiada para o arquivo especificado na seção de propriedades do Janela Propriedades do IBM Security Directory Integrator do projeto. O arquivo vinculado é configurado automaticamente quando você usa o comando **Arquivo** > **Abrir Configuração do Directory Integrator**.

Construtor de Projetos

Um construtor de projetos customizados está associado ao projeto cujo objetivo é montar todos os artefatos dentro de um *arquivo de configuração executável*.

Esse construtor pode ser executado automaticamente sempre que um recurso for alterado, ou manualmente por meio de um item de menu **Projeto** > **Construtor** padrão. De qualquer maneira, o construtor de projetos IBM Security Directory Integrator mantém um arquivo de configuração que é atualizado quando chamado. Quando um recurso for alterado (modificado, incluído ou removido), o construtor atualizará o arquivo de configuração executável com essa atualização de recursos. Apenas arquivos de configuração reconhecidos terão efeito; ou seja, AssemblyLines, mas não arquivos .gif, por exemplo. O arquivo de configuração executável fica normalmente oculto, uma vez que o nome do arquivo não começa com um ponto (.). O arquivo recebe o nome `.rs.xml`, localizado na pasta de projeto. Esse é o arquivo de configuração compilado que é enviado a um servidor IBM Security Directory Integrator para execução.

O construtor relocalizará e renomeará componentes da configuração de destino. Se a pasta Recursos contiver propriedades e conectores, eles serão relocalizados para suas pastas padrão da configuração de destino (ou seja, pasta Propriedades e Conectores).

O construtor de projetos verifica também todos os componentes modificados em relação a erros óbvios e possíveis problemas. Esses problemas são registrados na visualização **Problemas** do Eclipse padrão. Cada item de problema contém uma descrição do problema, bem como o local para que você possa dar um clique duplo e ativar o editor em que o problema foi identificado.

Propriedades e Substabelecimento

Todo projeto IBM Security Directory Integrator está associado a um servidor IBM Security Directory Integrator. O servidor associado é o servidor que será usado para executar os AssemblyLines no projeto. Como o servidor pode estar localizado em uma máquina diferente, o modelo de projeto deve fornecer acesso às propriedades por meio de cópias locais dos armazenamentos de propriedades escolhidos.

Os armazenamentos de propriedades podem ser transferidos por download do servidor, conforme for necessário, onde a cópia local contiver dois valores para cada propriedade. Um valor é aquele que foi transferido por download do servidor (valor remoto) e o outro é o valor definido pelo usuário (valor local). Isso é feito para que possam ser vistas as propriedades em possível conflito, bem como seja possível extrair o conjunto de propriedades em uso pela solução. Não há necessidade de transferir por download um armazenamento de propriedades antes de incluir propriedades nele; no entanto, quando você executa a solução, todas as propriedades com um valor local serão verificadas em relação ao valor no servidor para evitar substituições não intencionais de propriedades existentes.

A edição dos arquivos de propriedades pode ser feita abrindo-se (ou criando) os arquivos de propriedades na pasta Recursos. Depois de criado um arquivo de propriedade, você pode modificar seu conteúdo e fazer upload e download. Upload e download é o que você faz para sincronizar as propriedades localmente com aquelas no servidor.

Nota: O IBM Security Directory Integrator usa atualmente o sinal de igual "=" ou dois-pontos ":" como separador nos arquivos de propriedades de pares chave/valor, seja qual for o primeiro. O uso do sinal de igual ou de dois pontos em nomes de propriedades e valores de propriedades, portanto, não é suportado. O separador de chave/valor do arquivo de propriedades no IBM Security Directory Integrator V6.0 e anterior era apenas o caractere ":"; portanto, os arquivos de propriedades migrados da V6.0 e anterior podem requerer edição.

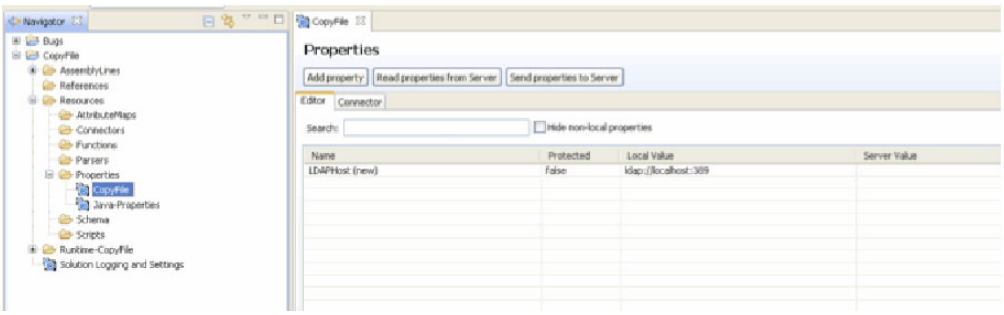


Figura 8. Visualização Propriedades

Observe que os armazenamentos de propriedades customizadas com um caminho relativo (por exemplo, a configuração do Conector usa um caminho relativo) terão um arquivo correspondente gerado no diretório *Project* do tempo de execução. Quando você cria novas propriedades customizadas, o caminho padrão será definido como "{config.$directory}/*Nome do Arquivo*.properties" em que *Nome do Arquivo* é o nome dado ao novo armazenamento de propriedades. O "{config.$directory}" é resolvido no local do arquivo de tempo de execução.

Por padrão, os armazenamentos de propriedades compartilhadas não são incluídos em um projeto (por exemplo, Armazenamento Global, de Soluções e do Sistema). Você poderá ainda incluí-los no projeto se desejar manter as alterações nesses arquivos. Para visualizar os armazenamentos de propriedades compartilhadas, você deve usar a visualização Servidores ou usar **Procurar Armazenamentos do Sistema** na barra de ferramentas principal.

O Modelo da Interface com o Usuário

A User Interface (UI) é fornecida por um conjunto de visualizações, editores e outros relacionados à UI. Eles são fornecidos por meio dos mecanismos de ponto de extensão padrão, conforme definido pela plataforma Eclipse.

Embora existam várias contribuições de ponto de extensão no CE, somente as mais importantes são listadas aqui.

Tabela 7. Contribuições de Ponto de Extensão do Eclipse CE

Contribuição	Descrição
Editores	Editores são fornecidos para todos os principais arquivos de configuração: • AssemblyLine (arquivos `.assemblyline`) • Conector (arquivos `.connector`) • Função (arquivos `.function`) • Scripts (arquivos `.script`) • Propriedades (arquivos `.tdiproperties`) • AttributeMap (arquivos `.attributemap`)
Visualizações	Várias visualizações são fornecidas para auxiliar na visualização de vários aspectos dos arquivos de configuração.
Menus, Barras de Ferramentas	Todas as ações no CE que operam nos objetos de configuração são definidas como ações padrão.
Assistentes	Vários assistentes são fornecidos para auxiliar na criação de novos projetos e arquivos de configuração.

O CE segue o paradigma MVC (Model, View, Controller). O editor para um arquivo de configuração cria os widgets que mostram o conteúdo do arquivo de configuração. Os widgets registram barras de ferramentas e menus que eles usam com a estrutura Eclipse, portanto contribuições podem ser feitas em relação a eles.

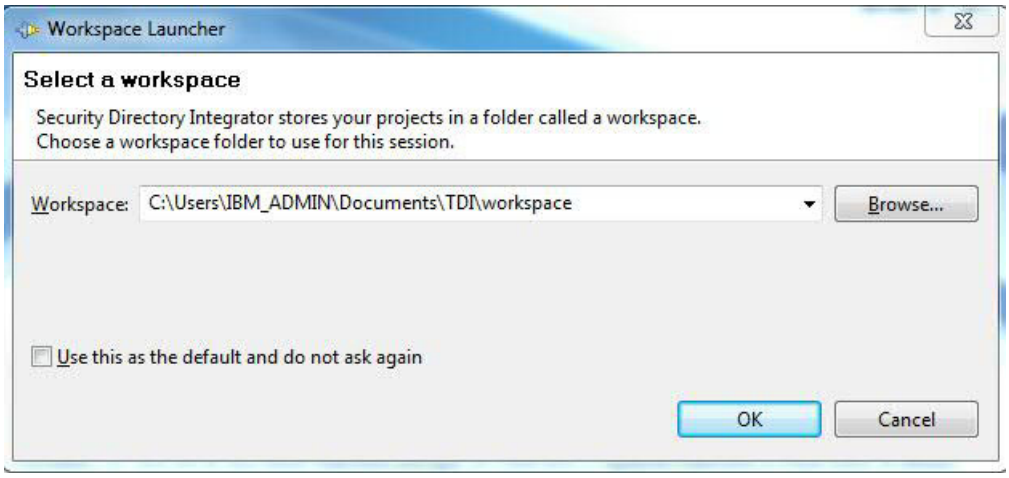
Todas as ações que afetam o arquivo de configuração são implementadas e contribuídas usando mecanismos padrão do Eclipse.

A Interface com o Usuário

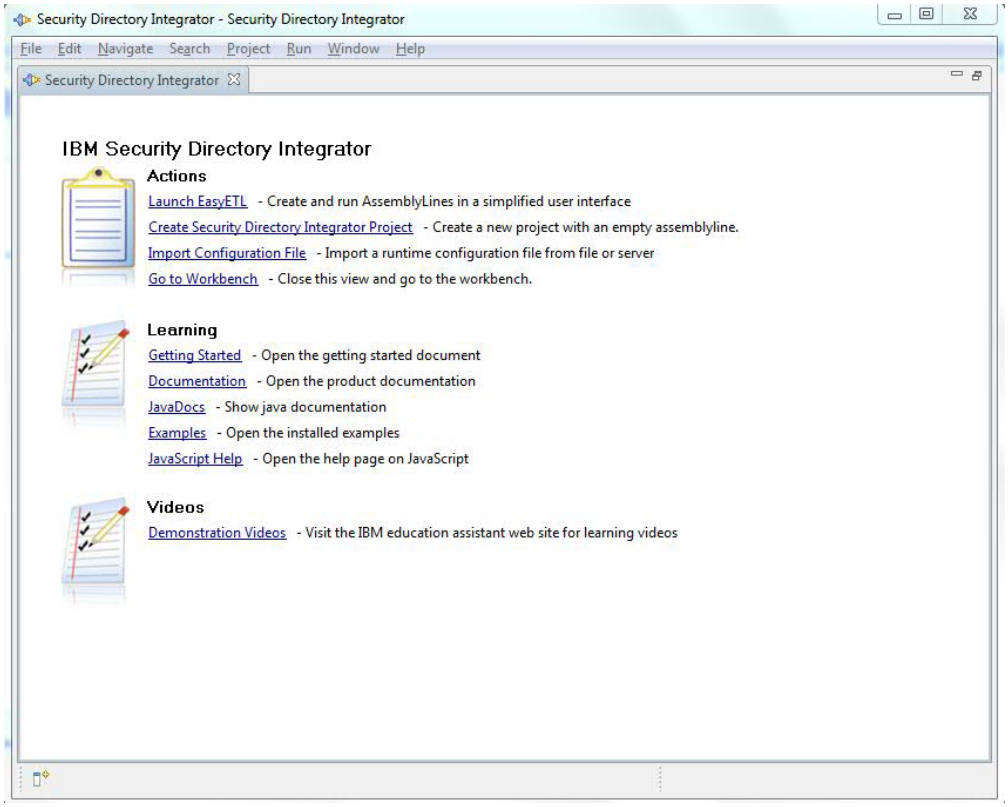
Janela de Aplicativo

Na primeira vez em que você iniciar o Editor de Config (CE), ele solicitará um diretório da área de trabalho. O diretório da área de trabalho é o local do sistema de arquivos onde seus projetos são armazenados. Você pode alterar esse diretório posteriormente no menu **Arquivo**.

Esse diálogo será mostrado sempre que você iniciar o CE, a menos que selecione a caixa de opções para tornar o diretório específico da área de trabalho o local padrão.

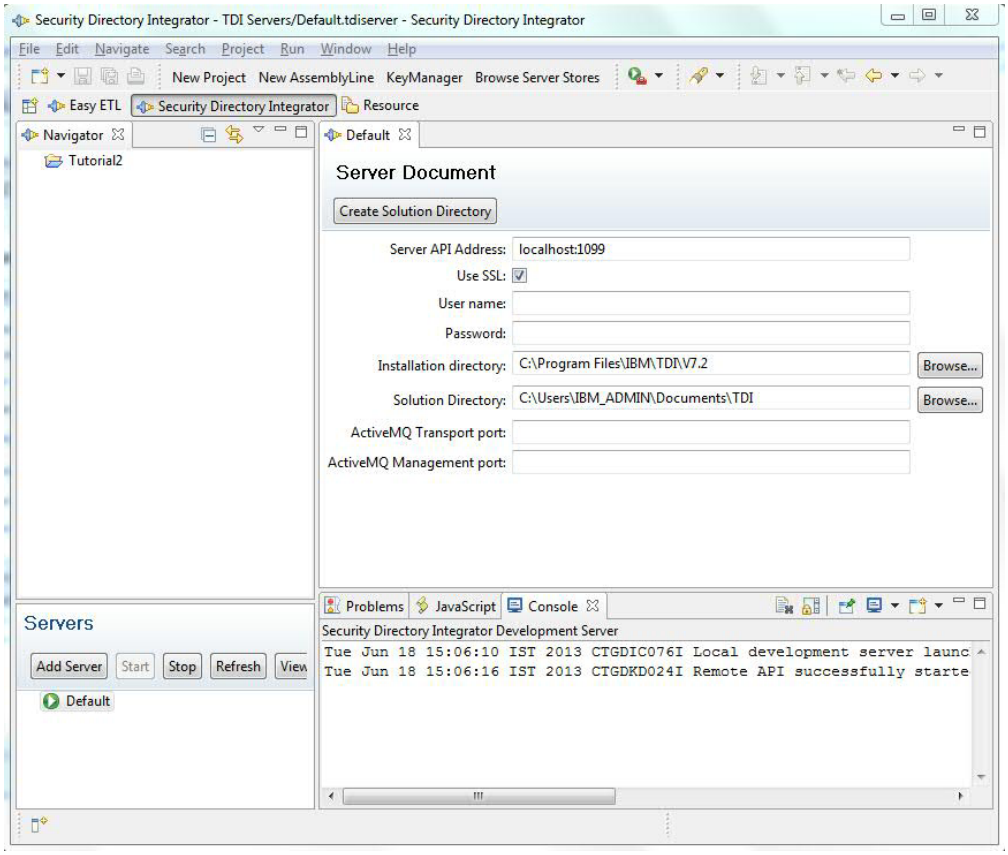


Após esse diálogo, a janela principal da área de trabalho será exibida. Se essa for a primeira vez em que você inicia o CE, ele mostrará a tela de boas-vindas:



Essa tela de boas-vindas fornece alguns links rápidos para tarefas comuns e sites de informações. O link da documentação o leva à documentação do produto configurado (propriedade do sistema *com.ibm.tdi.helpLoc*).

A tela de boas-vindas pode ser reaberta escolhendo **Ajuda** > **Bem-vindo**. Quando a tela de boas-vindas for fechada, você verá a janela da área de trabalho:



Essa é a janela principal, em que você gerencia seus projetos e configurações.

Nesta imagem, há um editor aberto (do documento Default.server) e várias visualizações. As visualizações mais importantes são aquelas que você vê nesta imagem.

O *Navegador* (parte superior à esquerda) contém todos os projetos e arquivos de origem para configurações do servidor e soluções do IBM Security Directory Integrator. O navegador pode conter também outros arquivos e projetos, como arquivos de texto, etc. O CE tratará os projetos IBM Security Directory Integrator especificamente, de modo que outros arquivos e projetos não sejam afetados pelo CE. Você verá como na seção sobre o construtor de projetos.

A visualização *Servidores* (parte inferior à esquerda) mostra o status de cada um dos servidores definidos no projeto "Servidores IBM Security Directory Integrator". Você pode ter quantos servidores definidos desejar. A visualização fornece várias funções para operar em servidores e suas configurações. O botão de atualização atualizará o status de todos os servidores da visualização.

A *área do editor* (parte superior à direita) é onde todos os editores são mostrados. Quando você abre um documento, como uma configuração do AssemblyLine, ele é aberto nesta área. Essa área é dividida verticalmente com uma área (parte inferior à direita) que contém várias visualizações para fornecer outras informações relevantes. Entre as mais importantes estão a visualização Problemas que mostra potenciais problemas com um componente do IBM Security Directory Integrator, o Log de Erros que mostra erros que ocorrem ao desenvolver soluções e, finalmente,

a visualização Console que mostra o log do console de servidores IBM Security Directory Integrator em execução (por exemplo, aqueles que são iniciados pelo CE).

Visualização Servidores

A Visualização Servidores contém várias funções úteis para gerenciar instâncias do servidor. Além de incluir e remover instâncias do servidor, há vários comandos associados aos servidores e suas instâncias de configuração e AssemblyLines ativos.

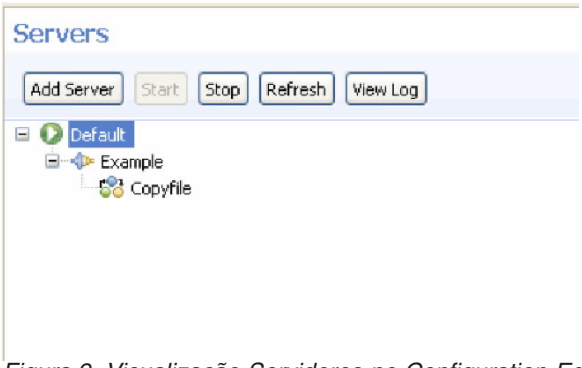


Figura 9. Visualização Servidores no Configuration Editor

A barra de ferramentas principal mostra os seguintes botões:

Incluir Servidor
Use-o para incluir um outro servidor

Iniciar Utilize para iniciar um servidor "local" (por exemplo, um que seja acessível em seu sistema de arquivos).

Se tiver uma instância de configuração selecionada, você poderá iniciar um ou mais de seus AssemblyLines.

Parar Utilize para enviar um pedido de parada para o servidor, instância de configuração ou AssemblyLine.

Atualizar
Use-o para atualizar o conteúdo da visualização Servidores

Visualizar Log
Use-o para visualizar o arquivo de log padrão, o arquivo "ibmdi.log" em um servidor "local"

O menu pop-up de cada item na visualização mostra comandos adicionais que podem ser executados com base na seleção.

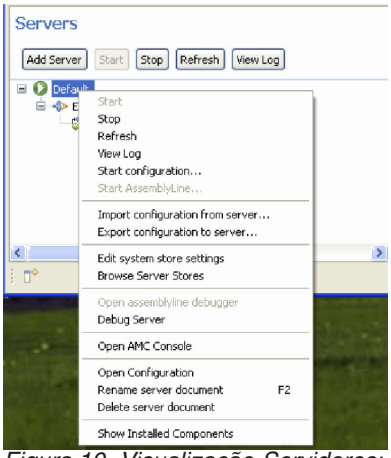


Figura 10. Visualização Servidores; Menu Pop-up

Os comandos possíveis são:

Iniciar Iniciar Servidor

Parar Para o servidor, instância de configuração ou AssemblyLine.

Atualizar
Atualiza a visualização de servidores.

Visualizar Log
Abre o arquivo ibmdi.log em um editor de texto

Iniciar Configuração...
Quando um servidor é selecionado, você pode iniciar uma instância de configuração nesse servidor.

Iniciar AssemblyLine....
Quando uma instância de configuração é selecionada, você pode iniciar um AssemblyLine a partir dessa instância de configuração.

Importar Configuração do Servidor...
Permite importar uma configuração do servidor em um projeto do CE.

Exportar Configuração para o Servidor...
Permite exportar um projeto do CE para uma configuração de tempo de execução no servidor selecionado.

Editar configurações do armazenamento do sistema
Abre as configurações do armazém do sistema para o servidor selecionado

Pesquisar Armazéns do Sistema
Abre o navegador de dados do armazém do sistema para visualizar/editar tabelas do armazém do sistema.

Abrir Depurador do AssemblyLine
Anexa um depurador ao AssemblyLine selecionado. Isso permitirá depurar um AssemblyLine já em execução.

Servidor de Depuração
Abre uma sessão de depuração do servidor para o servidor selecionado

Mostrar Componentes Instalados
Mostra uma lista de componentes instalados e suas versões para o servidor selecionado

Abrir console do AMC
Abre o console do AMC para o servidor selecionado. Se o servidor foi instalado sem o AMC, esta opção estará esmaecida.

Nota: O recurso AMC foi descontinuado e será removido em uma versão futura do IBM Security Directory Integrator.

Excluir Documento do Servidor
Exclui o servidor da visualização de servidores

Renomear Documento do Servidor
Renomeia o servidor selecionado. Isso não tem efeito sobre o próprio servidor; é somente uma representação local do servidor.

As opções de menu são desativadas e ativadas com base na seleção.

Editor de Expressão

O editor de expressão está disponível em vários contextos diferentes. Geralmente, ao especificar valores de parâmetros, como parâmetros de conexão, critérios de vinculação, etc., você pode usar o editor de expressão em vez de digitar um único valor para o parâmetro.

Propriedade de Uso

Essa opção permite escolher uma propriedade existente de seus armazenamentos de propriedade ou criar um novo par propriedade/valor.

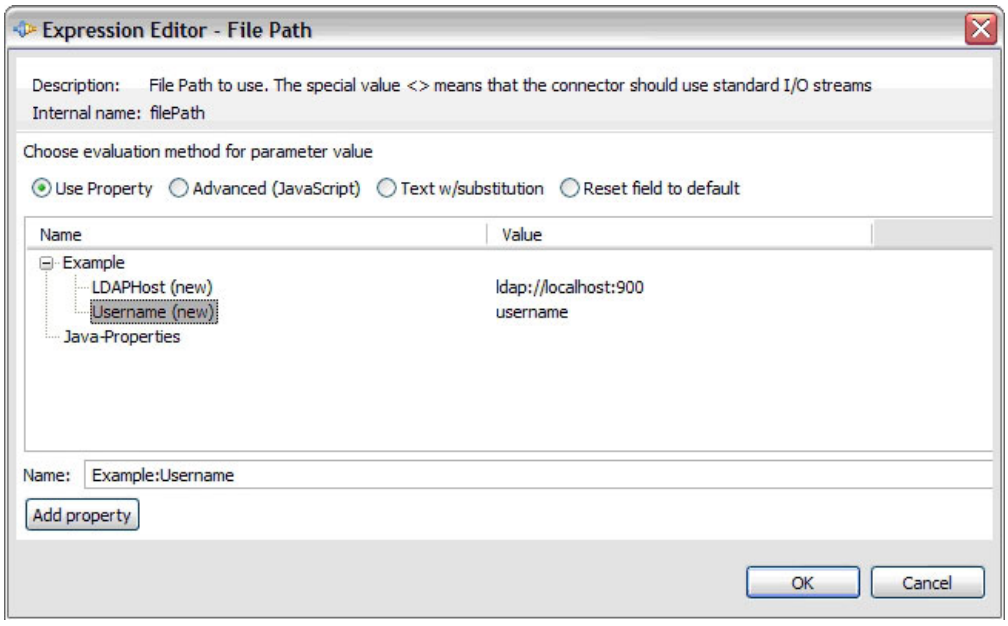


Figura 11. Editor de Expressão: Propriedade Simples

Quando você seleciona uma propriedade, o campo de texto *Name* abaixo da árvore é atualizado com a expressão dessa propriedade. Por padrão, a expressão inclui o nome do armazenamento ("Exemplo" neste caso), seguido por dois-pontos e o nome da propriedade ("Nome do usuário" neste caso). Quando você clica em **OK**, a expressão no campo de texto é usada para o parâmetro. Isso significa também que você pode digitar a expressão diretamente neste campo, sem passar pela árvore de propriedades. Se você, por exemplo, souber que há uma propriedade

chamada "FilePath" no servidor, essa solução será executada; você poderá remover o nome do armazenamento se não o conhecer (ou seja, digite "FilePath" sem o prefixo "store:").

Avançado (JavaScript)

Com essa opção, o valor é computado como resultado do código JavaScript que você insere no editor.

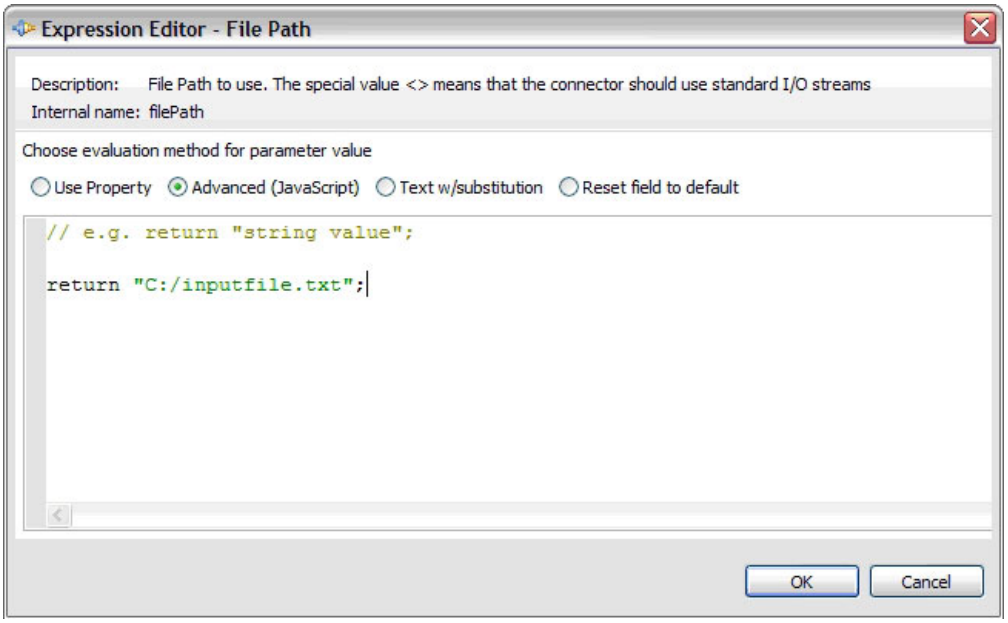


Figura 12. Editor de Expressão: Avançado (JavaScript)

Texto com Substabelecimento

Esta é a "Expressão do IBM Security Directory Integrator" da 6.x. A partir da versão 7, recomenda-se usar JavaScript para avaliação complexa de variáveis e propriedades. Essa opção, no entanto, é muito útil quando você deseja inserir grandes quantidades de texto. As opções de substabelecimento são compatíveis com as expressões da versão 6.

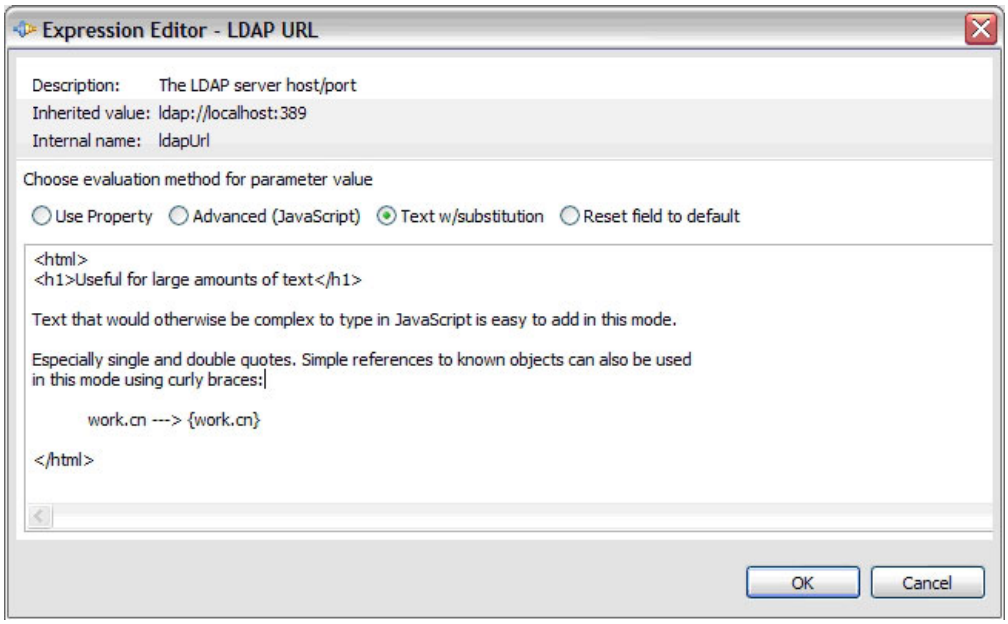


Figura 13. Editor de Expressão: Texto com Substabelecimento de Estilo da v.6

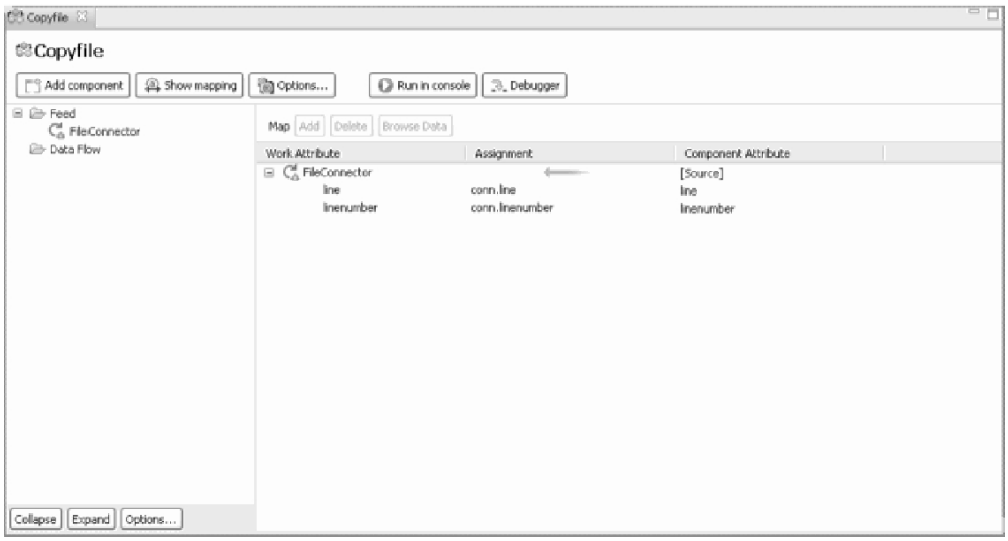
Redefinir Campo como Padrão

A última opção é usada para redefinir o valor de parâmetro como seu valor padrão (também conhecido como o valor herdado). Selecione essa opção e pressione **OK** para redefinir o valor de parâmetro.

Editor de AssemblyLine

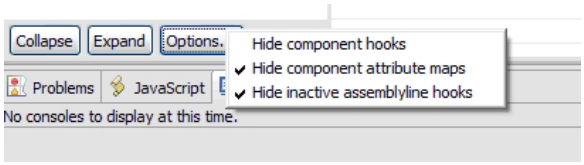
O editor de AssemblyLine é o editor principal usado ao desenvolver soluções do IBM Security Directory Integrator.

Nesse editor, você compila o AssemblyLine incluindo e configurando componentes. À medida que você avança, poderá executar o AssemblyLine para ver os efeitos dos componentes incluídos.

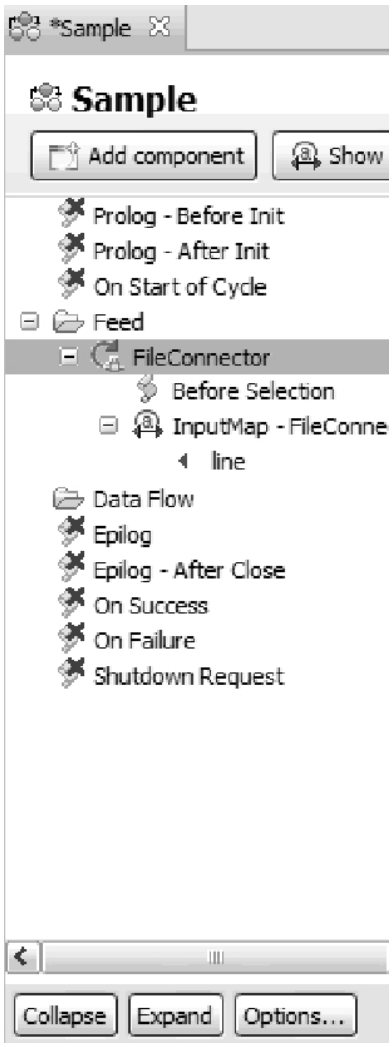


O Editor de AssemblyLine mostra duas seções principais. A seção à esquerda mostra os componentes do AssemblyLine e os ganchos. Na barra de ferramentas,

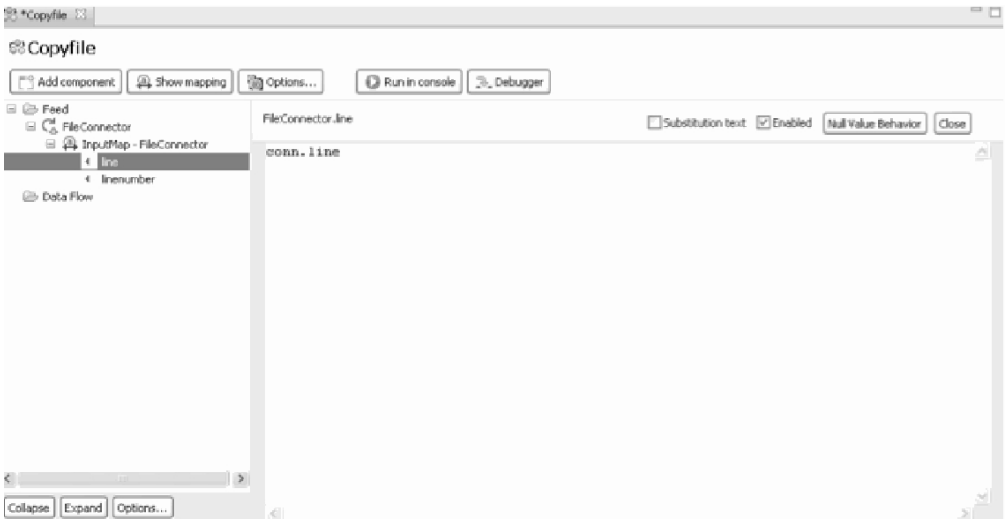
você pode escolher o nível de detalhes que deseja ver na árvore.



O botão **Opções...** permite escolher quanto do AssemblyLine será revelado na visualização em árvore do componente.

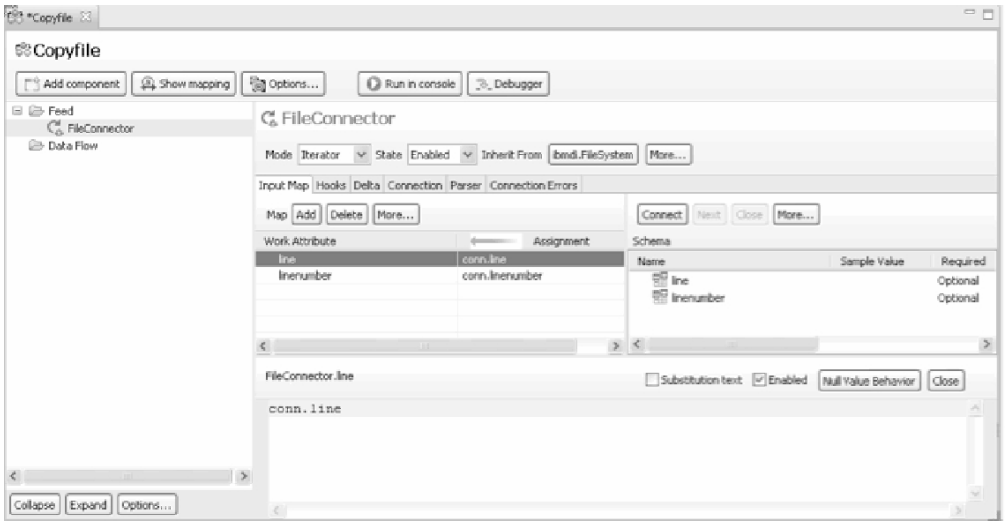


Essa imagem mostra todos os ganchos do AssemblyLine, bem como mapas de atributos do componente na árvore. Quando você seleciona ganchos ou itens do mapa de atributos nessa árvore, o lado direito fornecerá uma visualização maior do item dentro do próprio editor do componente:



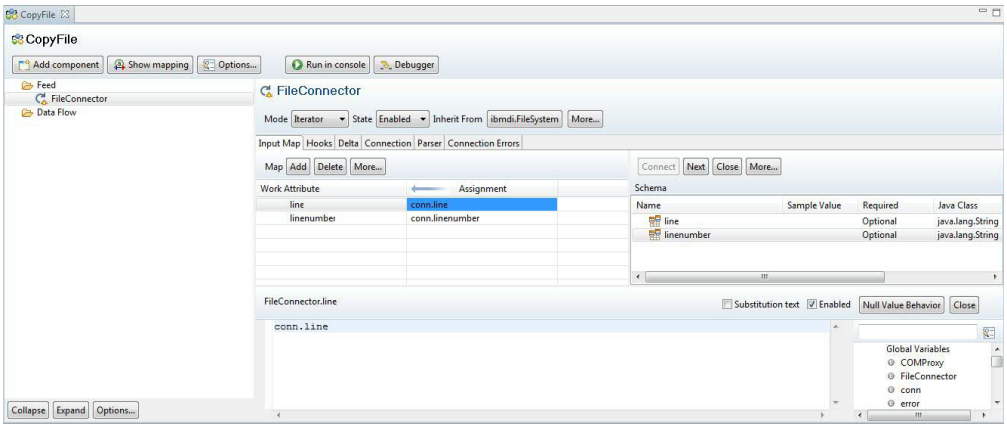
A seção de mapeamento à direita mostra os mapas de todos os componentes com um mapa de atributos. O primeiro nível desta árvore é o nome do componente com os itens individuais do mapa de atributos abaixo deles. Selecionar esse item abre o editor de detalhes desse item. Se você mostrar ganchos na visualização de componentes do AssemblyLine, poderá dar um clique duplo neles, bem como abrir o editor de scripts.

A imagem abaixo mostra como o editor rápido aparece com o script para o mapa de atributos.

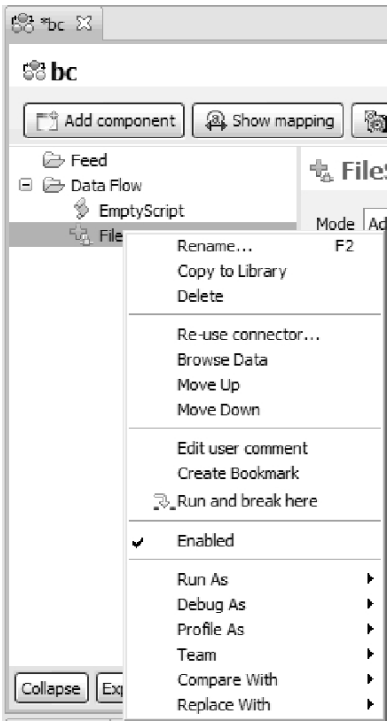


O editor rápido mostra também algumas opções adicionais. O "Texto de Substituição" é o formato "Expressão do IBM Security Directory Integrator" versão 6, em que é possível inserir texto e algumas macros de expansão simples. Na versão 7, esse formato é basicamente destinado a valores de texto grandes ou complexos, uma vez que o JavaScript fornece uma sintaxe de expressão mais poderosa.

A seção de fluxo de componentes mostra todos os componentes do AssemblyLine. Quando você seleciona um componente, o lado direito do editor é substituído pela tela de configuração desse componente. Um atalho útil na CE é Ctrl-M que maximiza o editor ou a visualização atual para preencher a tela do aplicativo. Ctrl-M alternará entre o tamanho normal e maximizado.



Quando você clica em um componente, uma tela de configuração desse componente substitui a visualização geral de mapeamento de atributo. Também é possível clicar com o botão direito do mouse no componente para acessar o menu pop-up do componente.



Opções do AssemblyLine

No botão suspenso **Configurações**, você pode selecionar várias opções.

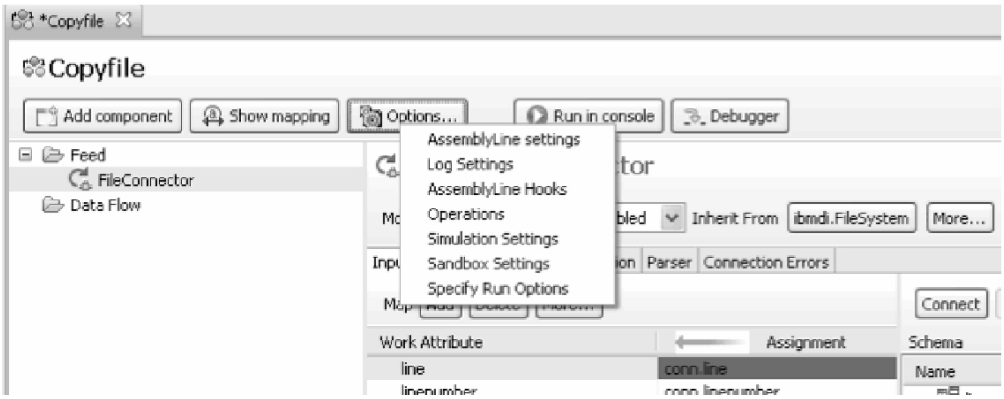


Figura 14. Menu de Opções do AssemblyLine

Há várias telas de opções disponíveis no menu suspenso, que controlam diversos aspectos do AssemblyLine em termos de design e de opções de tempo de execução. Essas telas são:

- "Configurações do AssemblyLine"
- "Configurações do Log" na página 96
- "Ganchos do AssemblyLine" na página 97
- "Operações do AssemblyLine" na página 98
- "Configurações de Simulação" na página 99
- "Configurações da Sandbox" na página 100

Configurações do AssemblyLine

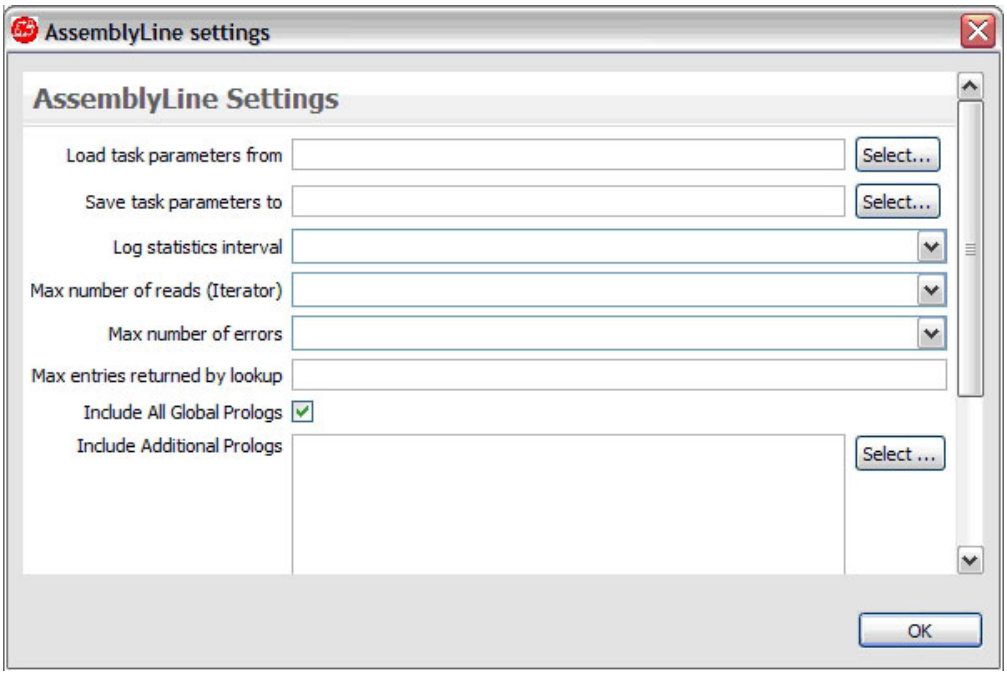


Figura 15. Configurações do AssemblyLine

Nessa janela você pode especificar opções que afetam como o AssemblyLine é executado.

Configurações do Log

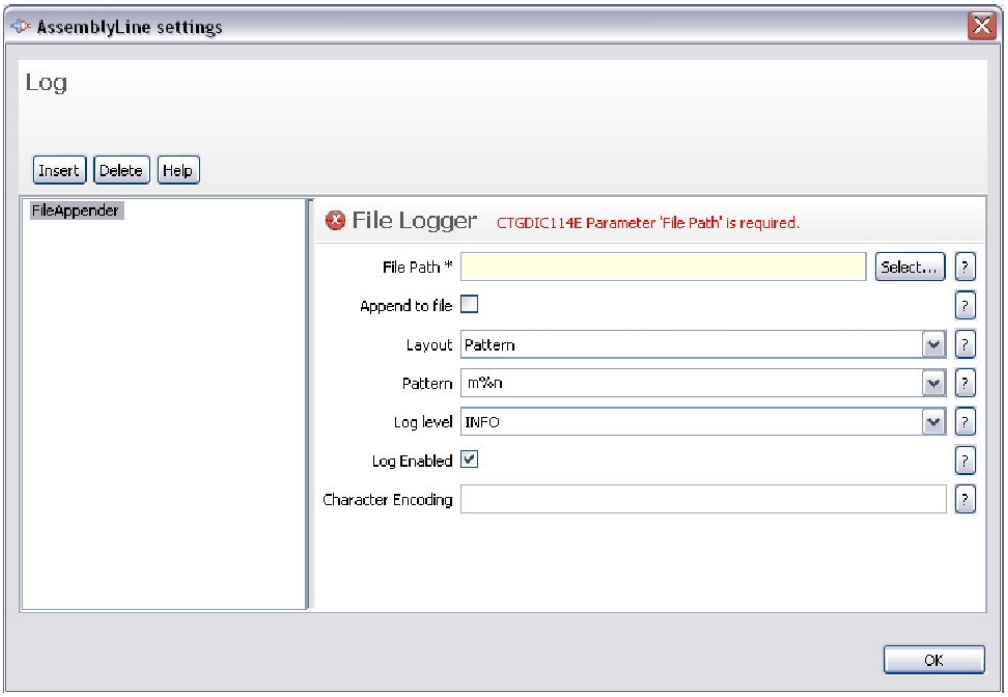


Figura 16. Configurações do Log do AssemblyLine

Nessa janela você pode incluir criadores de logs para este AssemblyLine. Os criadores de logs não são globais e são ativados apenas no AssemblyLine.

Ganchos do AssemblyLine

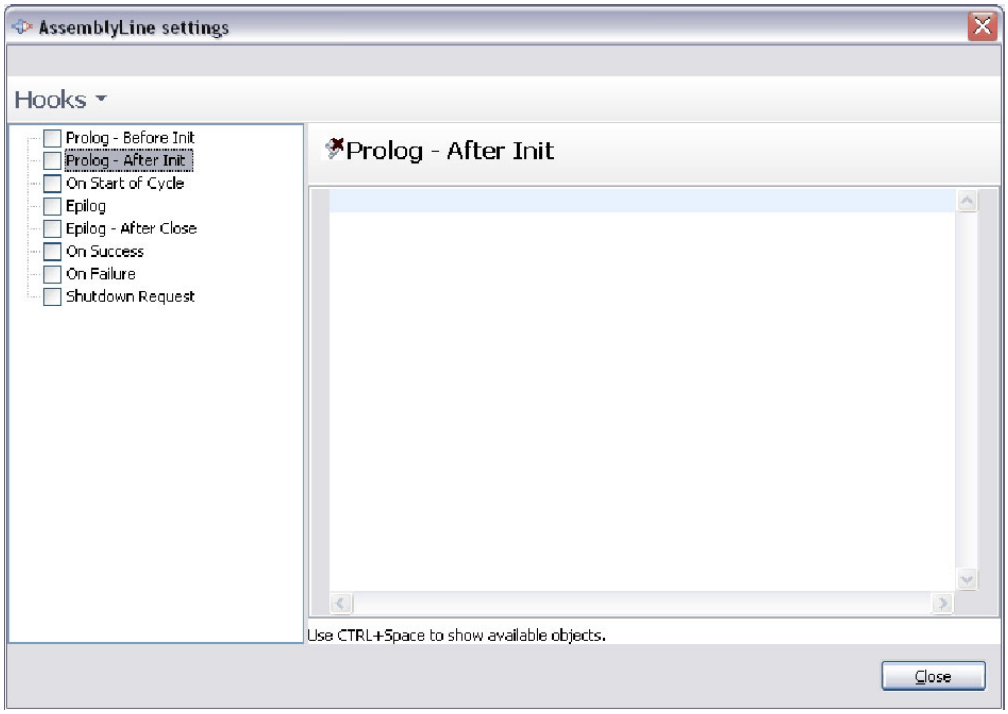


Figura 17. Ganchos do AssemblyLine

Nessa janela você pode ativar/desativar os ganchos no nível do AssemblyLine. Os ganhos ativados são mostrados também no painel de componentes do Editor de AssemblyLine.

Operações do AssemblyLine

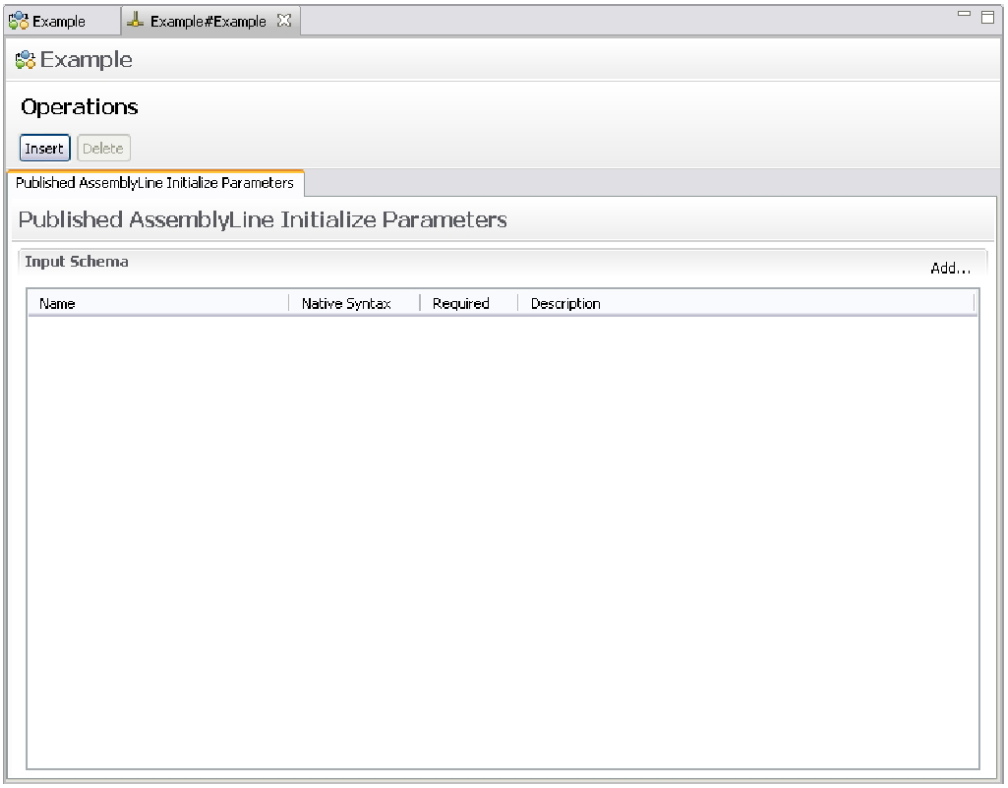


Figura 18. Operações do AssemblyLine

Essa janela permite definir operações do AssemblyLine. Consulte a seção Operações do AL, em "Criando Novos Componentes Usando Adaptadores" no *Referência* para obter uma descrição completa das operações. O botão Inserir permite inserir uma nova operação. Os **Parâmetros de Inicialização do AssemblyLine Publicados**, disponíveis por padrão, representam uma operação especial que é usada para fornecer valores para um AssemblyLine antes de os componentes serem inicializados.

Configurações de Simulação

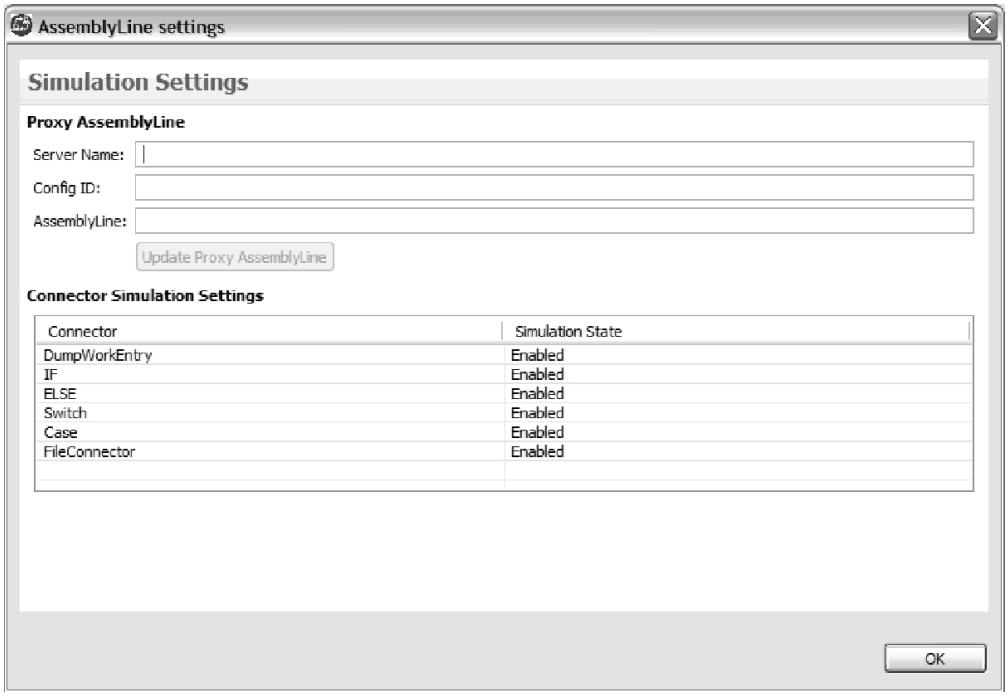
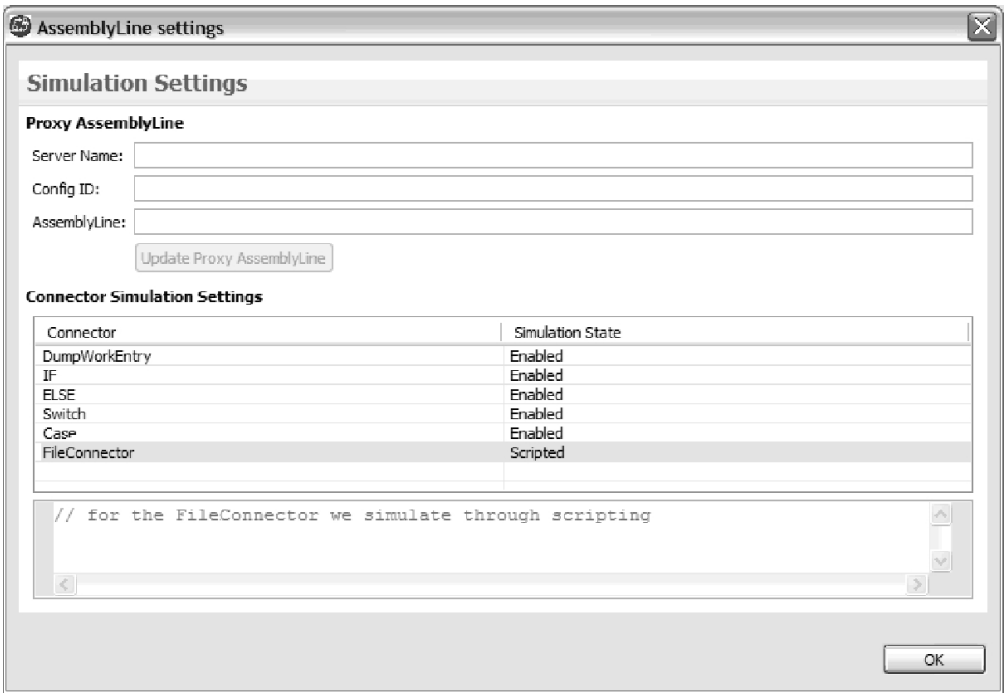


Figura 19. Configurações de Simulação do AssemblyLine

Essa janela permite definir as configurações de simulação por componente. Consulte “Modo de Simulação do AssemblyLine” na página 187 para obter informações adicionais. O painel mostrará um editor de scripts na parte inferior quando você optar por criar um script para simulação:



Você pode também criar/atualizar o AssemblyLine do proxy usado pelo código de simulação com o botão **Atualizar AssemblyLine do Proxy**.

Figura 20. Janela Configurações de Simulação do AssemblyLine, com Editor de Scripts

Configurações da Sandbox

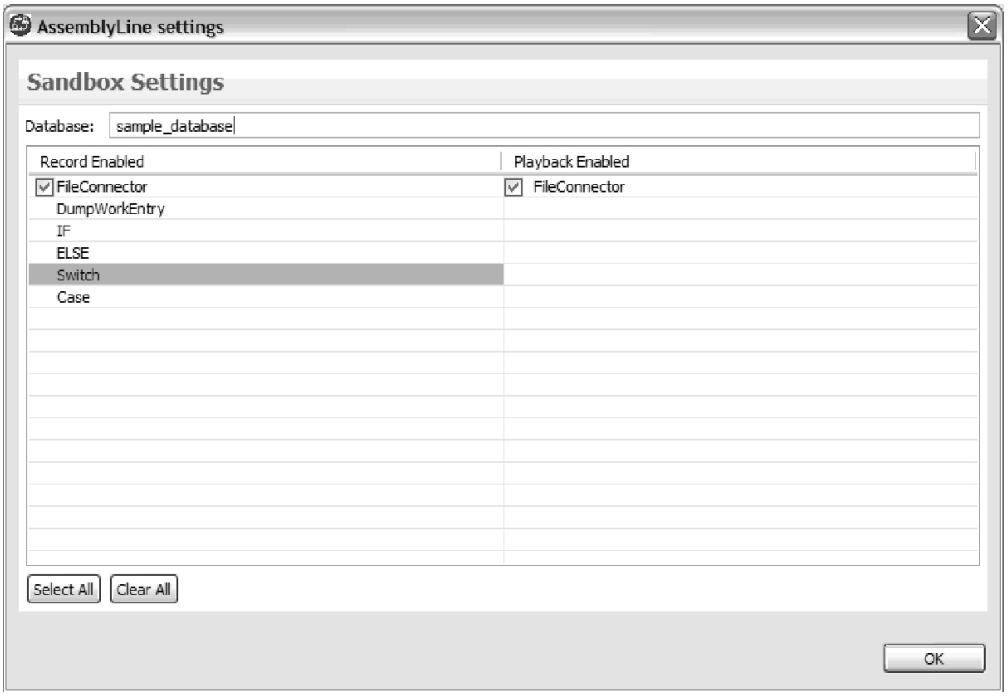


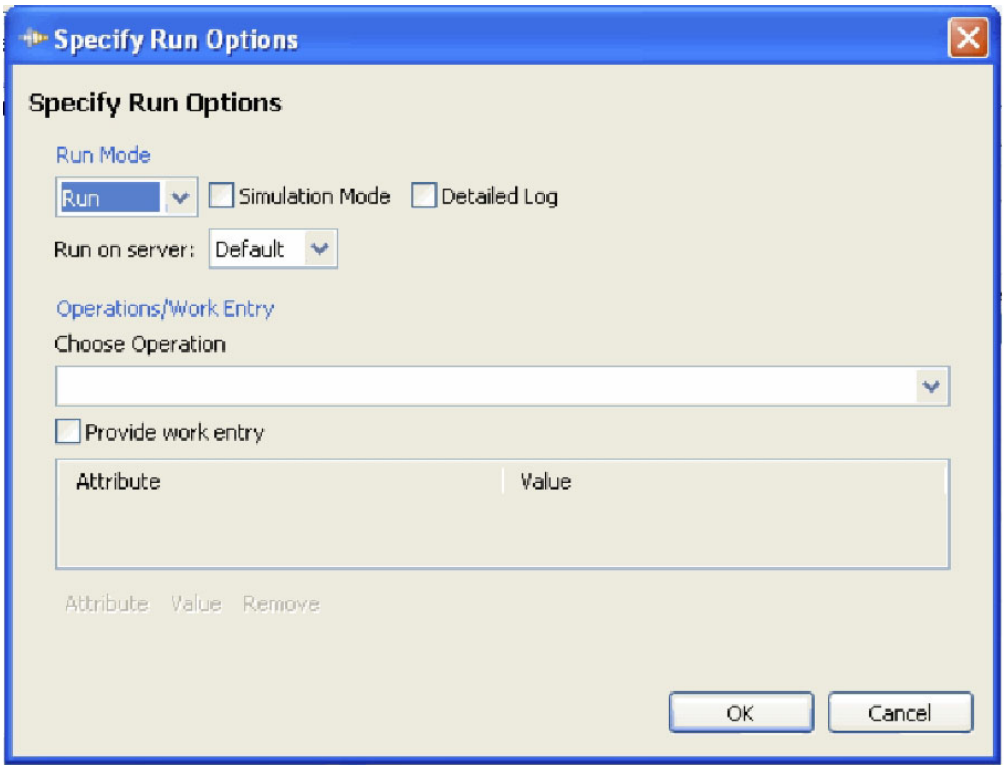
Figura 21. Configurações da Sandbox do AssemblyLine

As configurações da sandbox permitem que você configure os componentes que são gravados/reproduzidos ao executar o seu AssemblyLine no modo Gravar ou Reproduzir.

Para obter informações adicionais sobre a funcionalidade Ambiente de Simulação no IBM Security Directory Integrator, consulte “Sandbox” na página 185.

Especificar Opções de Execução

O diálogo de opções de execução o permite configurar como o AssemblyLine é executado.



Quando você seleciona **Fornecer Entrada de Trabalho**, pode construir uma Entrada estática que é alimentada no AssemblyLine quando ele é iniciado.

Figura 22. Diálogo Especificar Opções de Execução

Painéis de Componentes

Quando você abre um componente do AssemblyLine, obterá um painel de editor rápido na parte inferior da janela AssemblyLine.

Você o obterá para os seguintes componentes:

- “Ramificação IF/ELSE/ELSE-IF” na página 102
- “Ramificação Switch/Case” na página 102
- “Valor do Atributo For-Each” na página 104
- “Loop Condicional” na página 104
- “Loop do Conector” na página 105
- “Mapa de Atributos” na página 106

Ramificação IF/ELSE/ELSE-IF

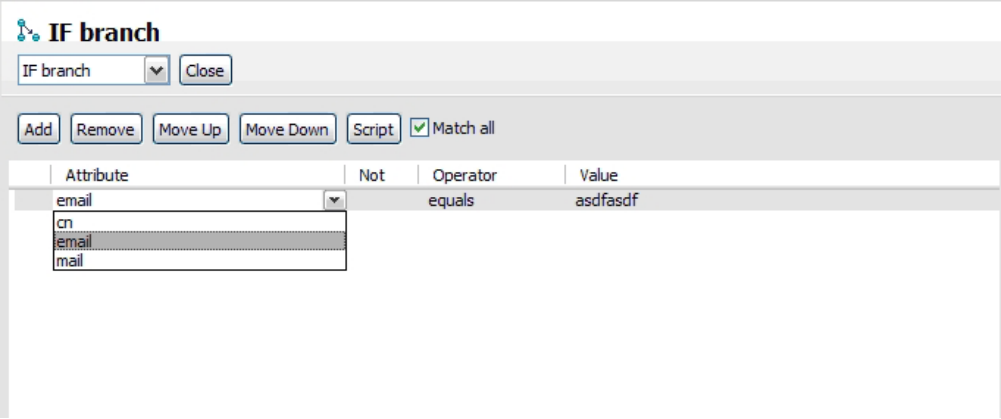


Figura 23. Editor Rápido para a Ramificação IF/ELSE/ELSE-IF

A ramificação IF e ELSE-IF permite que você especifique expressões simples e um script customizado que retorna *true* ou *false*. Use a caixa de opção **Corresponder todos** para retornar true apenas se todas as condições forem avaliadas como *true* (modo *AND*). Se estiver desmarcada, apenas uma das condições deverá corresponder (modo *OR*).

A ramificação ELSE não tem parâmetros.

Use o botão **Incluir** para incluir novas linhas de controles de Atributo/Operador/Valor. Você pode remover essas linhas usando o botão **Excluir**. O valor pode ser uma constante ou uma expressão. Use o editor de expressões para configurar a expressão clicando no botão após o campo de texto de valor. Os botões de movimento são usados para reorganizar as expressões. As expressões são avaliadas de cima para baixo. Você pode também alterar o tipo de ramificação (ou seja, IF, ELSE, etc.) usando o drop-down abaixo do título.

Ramificação Switch/Case

A configuração Switch tem várias opções para selecionar um valor. Esse valor é usado para corresponder os valores nas ramificações Case contidas. Quando elas forem iguais, a ramificação Case será executada. A ramificação Switch é sempre executada.

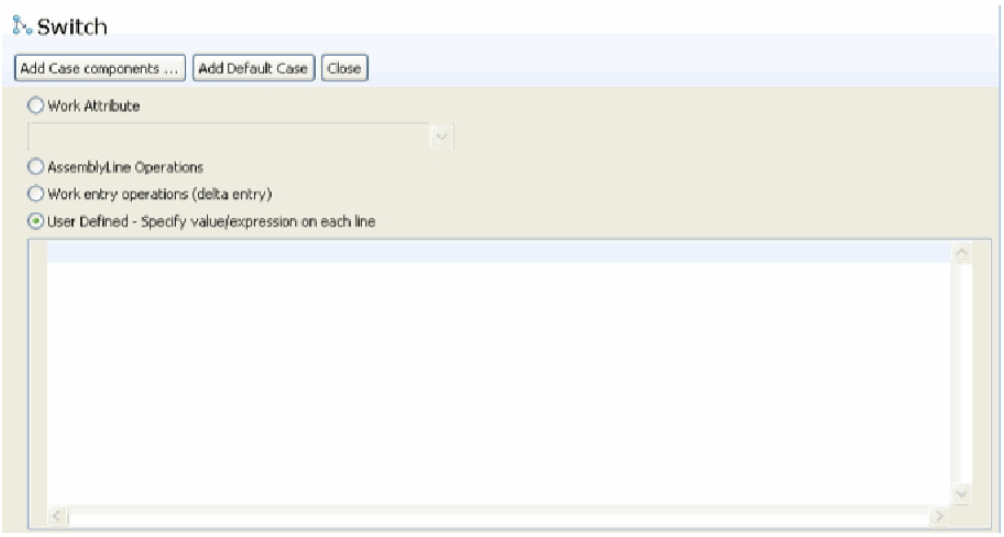


Figura 24. Ramificação Switch/Case

As opções de seleção que você pode escolher para a sua ramificação Switch/Case são:

Atributo de Trabalho
Selecione na lista de atributos de trabalho conhecidos

Operações do AssemblyLine
Selecione na lista de nomes de operações do AssemblyLine conhecidos.

Operações da Entrada de Trabalho
Usa o valor da operação da entrada de trabalho (por exemplo, método `work.getOperation()`).

Definido pelo Usuário
Aqui, você pode especificar o seu próprio valor.

Use o diálogo **Incluir Componentes Case** para gerar as ramificações Case dentro desta ramificação Switch. Com base em sua seleção, serão sugeridos automaticamente valores que fazem sentido para a seleção. Se você escolher operações de entrada de trabalho, serão sugeridos todos os valores de operação conhecidos (por exemplo, incluir, modificar).

Use o botão **Incluir Case Padrão** para incluir um case padrão. O case padrão será executado no caso de nenhuma das outras ramificações de case corresponderem ao valor do componente Switch.

Valor do Atributo For-Each

Figura 25. Loop de Valor do Atributo

Esse componente faz o loop sobre os valores em um atributo. Especifique o nome do atributo da entrada de trabalho sobre o qual fazer o loop (nome do atributo de trabalho) e o nome do atributo do loop. O nome do atributo do loop é definido como o valor do loop atual a partir do atributo de entrada de trabalho (por exemplo, `foreach f in work.attr.values; set loopattr = f`).

Loop Condicional

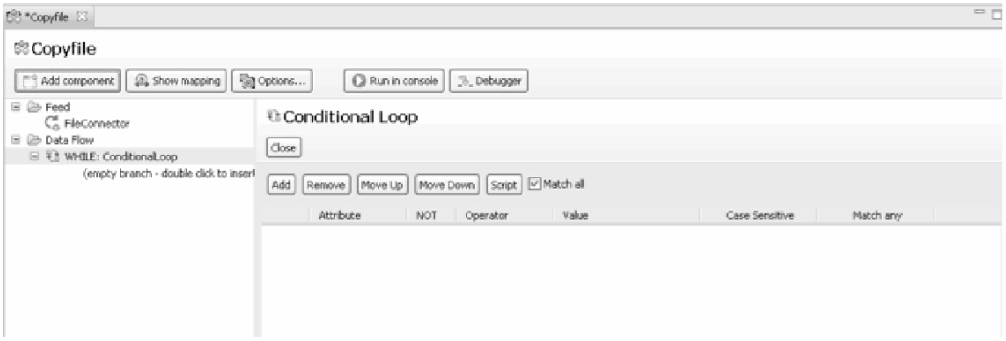


Figura 26. Loop Condicional

O loop condicional permite especificar expressões simples e um script customizado que retorna true/false.

Use o botão **Incluir** para incluir novas linhas de controles de Atributo/Operador/Valor. Você pode remover essas linhas usando o botão Excluir. O valor pode ser uma constante ou uma expressão. Use o editor de expressões para configurar a expressão clicando no botão após o campo de texto de valor.

A caixa de opção **Corresponder Todos** determina se todas as linhas devem corresponder (**Corresponder todos** marcada) ou se apenas as necessárias devem ser true para que a ramificação seja executada.

Loop do Conector

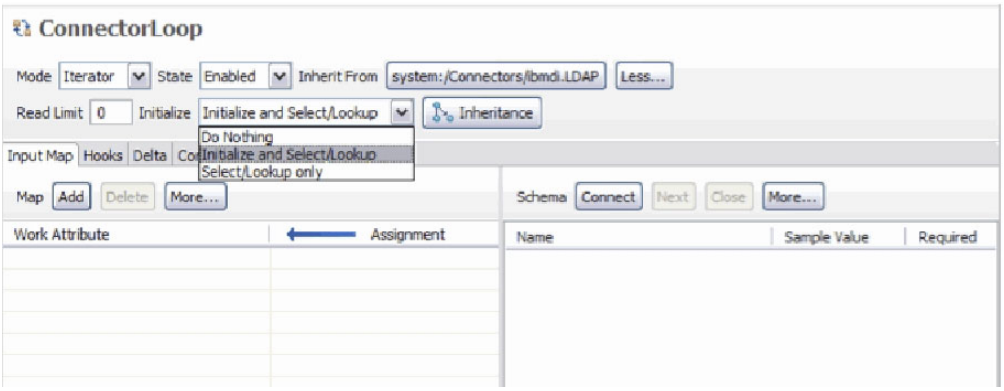


Figura 27. Loop do Conector

O Loop do Conector usa o editor do Conector para configurar um conector. Há algumas diferenças, conforme mostrado na imagem acima. A opção de inicialização mostra agora as opções relevantes para um loop do conector em vez das opções de inicialização normais. Além disso, apenas o Repetidor e a Consulta podem ser selecionados no drop-down Modo.

Com exceção das guias normais que você vê para um conector, há também um mapa de atributos de saída que permite configurar designações dinâmicas de parâmetros do conector na guia **Parâmetros do Conector**. Isso é ligeiramente diferente do mapa de saída comum no qual é mostrado um esquema corrigido, que é a lista de parâmetros do conector escolhido. A parte do esquema não possui os botões Conectar/Próximo para afetar o esquema.

Figura 28. Parâmetros do Conector no Loop do Conector

Mapa de Atributos

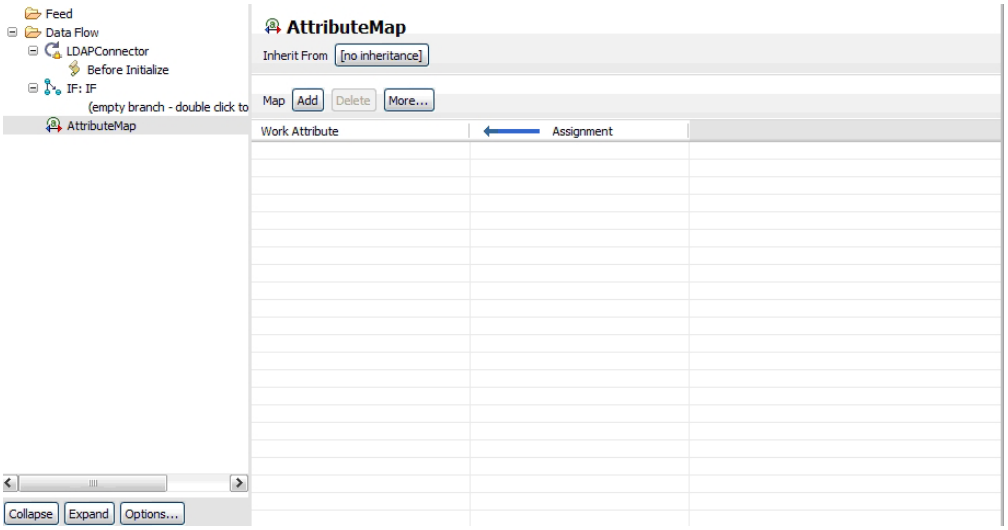


Figura 29. Componente do Mapa de Atributos Independente

O componente do mapa de atributos mostra um único painel em que você pode mapear atributos para a entrada de trabalho, independentemente do mapeamento de atributos dentro de outros componentes como um conector. O componente oferece também a reutilização de mapas de atributos de outros componentes, como conectores, funções e outros componentes do mapa de atributos em sua biblioteca. Para obter informações adicionais, consulte “Mapeamento de Atributos e Esquema” na página 109.

Visualização da Documentação do Usuário

Às vezes um AssemblyLine se transforma em algo totalmente complexo. Para ajudar na leitura da configuração, você pode documentar partes do AssemblyLine utilizando a visualização de documentação.

No esboço do AssemblyLine, você pode clicar com o botão direito do mouse em um componente e escolher **Editar Comentário do Usuário** para trazer a visualização da documentação para frente:

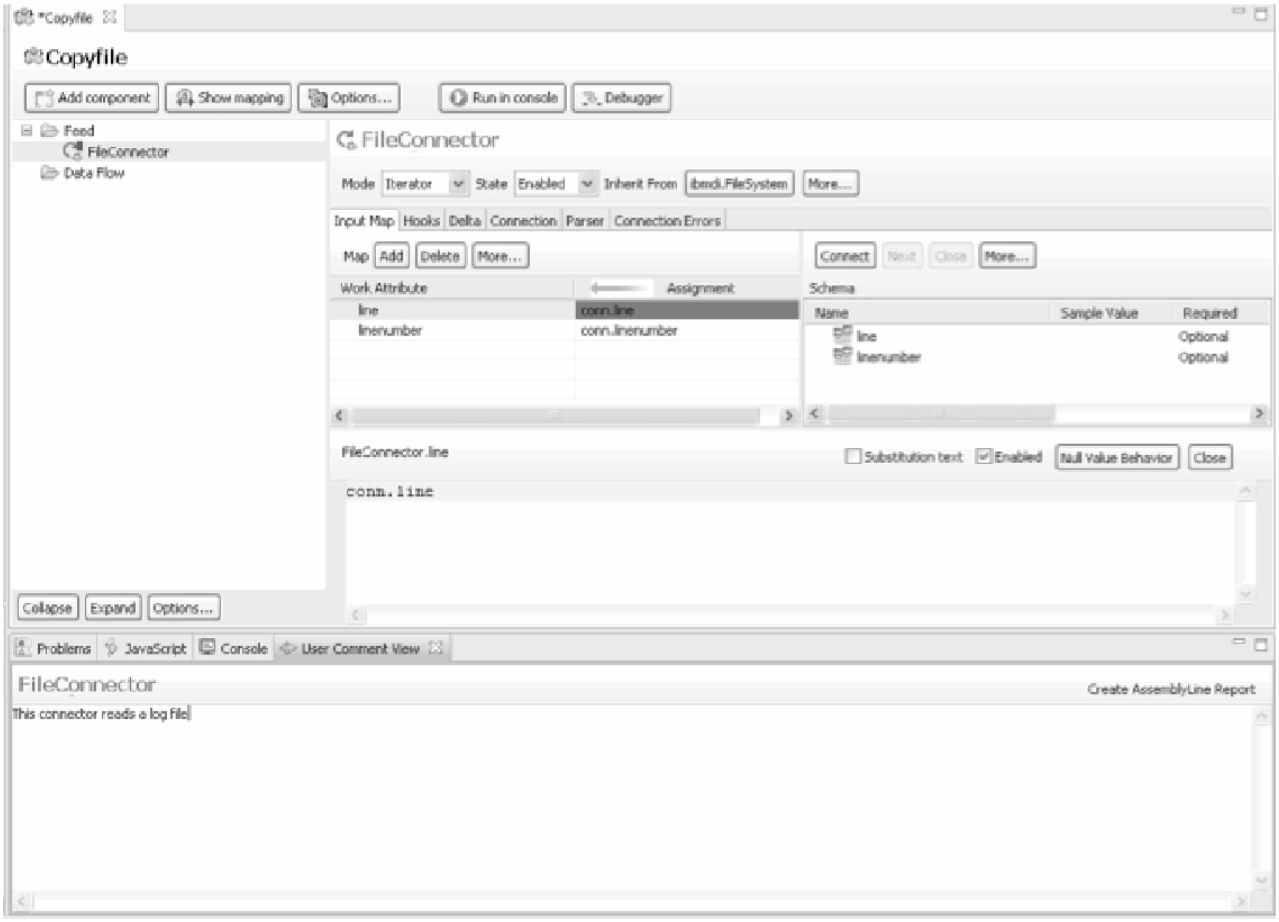


Figura 30. Visualização da Documentação do Usuário

Como a seleção muda no editor do AssemblyLine, a visualização da documentação reflete a seleção atual. O texto que você digita na visualização é salvo quando você salva o AssemblyLine. Componentes com um comentário são decorados no canto superior esquerdo do ícone do componente.

O botão **Criar Relatório do AssemblyLine** criará um relatório com todos os comentários do usuário inseridos no AssemblyLine. O modelo de relatório utilizado é chamado `UserCommentsReport.xsl` no diretório *TDI_Install_dir*`/XSLT/ConfigReports`.

Figura 31. Relatório do AssemblyLine de Amostra

Janela Executar AssemblyLine

Quando você executa o AssemblyLine, obterá uma janela com a saída do log mostrada.

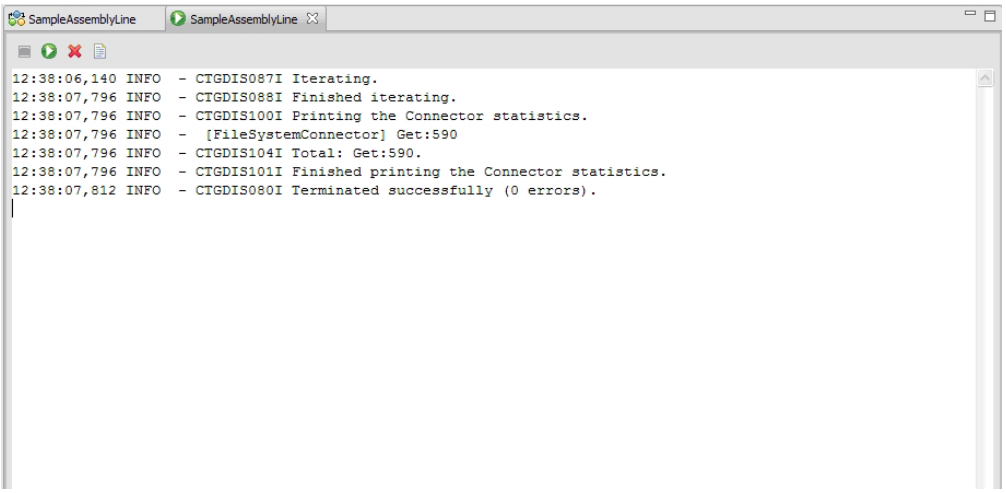


Figura 32. Log do Console

Nesta tela você pode também parar uma execução do AssemblyLine e reiniciá-la após sua conclusão.

Nota: Parar um AssemblyLine significa que o Editor de Config envia uma notificação de parada à Instância de Config (geralmente o Servidor local padrão) que está executando o AL; ele não cancela imediatamente o encadeamento, mas para a execução assim que o servidor recupera o controle. Isso difere das versões anteriores, em que pressionar o botão **Parar** faria com que o processo inteiro do servidor que estava executando o AssemblyLine fosse cancelado.
Os dois outros botões são para limpar a janela de log e abrir o arquivo de log em uma janela separada do editor. A janela de log mostra apenas as últimas centenas de linhas para evitar problemas de falta de memória.

O log é gravado em um arquivo temporário com um prefixo arquivo"tdi_ce_al_log" e uma extensão ".log". O arquivo é colocado no diretório temporário específico da plataforma, que é geralmente definido pela variável de ambiente TEMP/TMP. O arquivo de log é excluído automaticamente quando você fecha a janela Executar AssemblyLine, mas no caso de o aplicativo ou a máquina travar, você pode ter de remover manualmente esses arquivos de log. O editor para usar esse arquivo é padronizado como o editor de texto simples, mas pode ser alterado mapeando a extensão ".log" para um editor diferente (incluindo editores externos). Use a opção de menu **Janelas** > **Preferências** para abrir o seguinte diálogo:

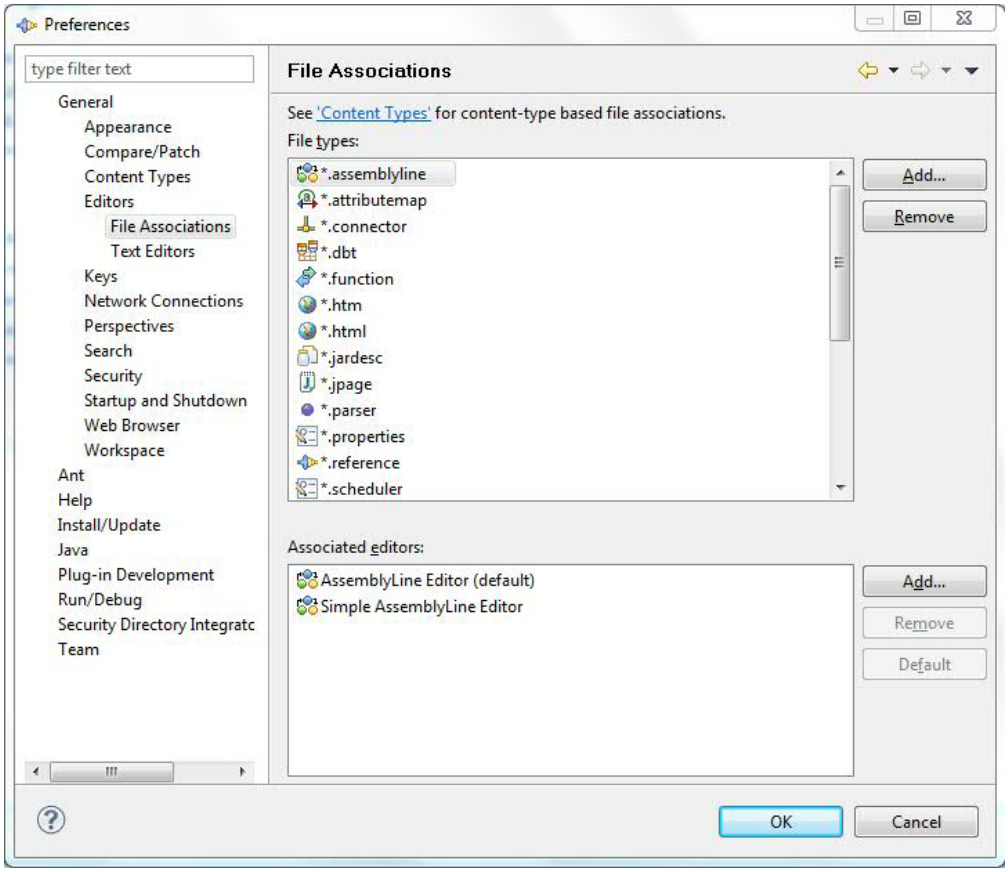


Figura 33. Preferências de Associações de Arquivos do Editor de Configuração

Aqui você pode incluir a extensão ".log" e associá-la a um editor.

Mapeamento de Atributos e Esquema

O Mapeamento de atributos é feito usando o painel de mapas de atributos do AssemblyLine ou do editor de componentes.

No Editor de AssemblyLine, você pode incluir atributos clicando com o botão direito do mouse na seção de mapas de atributos e optando por incluir um atributo ou utilizar o botão **Incluir** na barra de ferramentas, conforme mostrado abaixo.

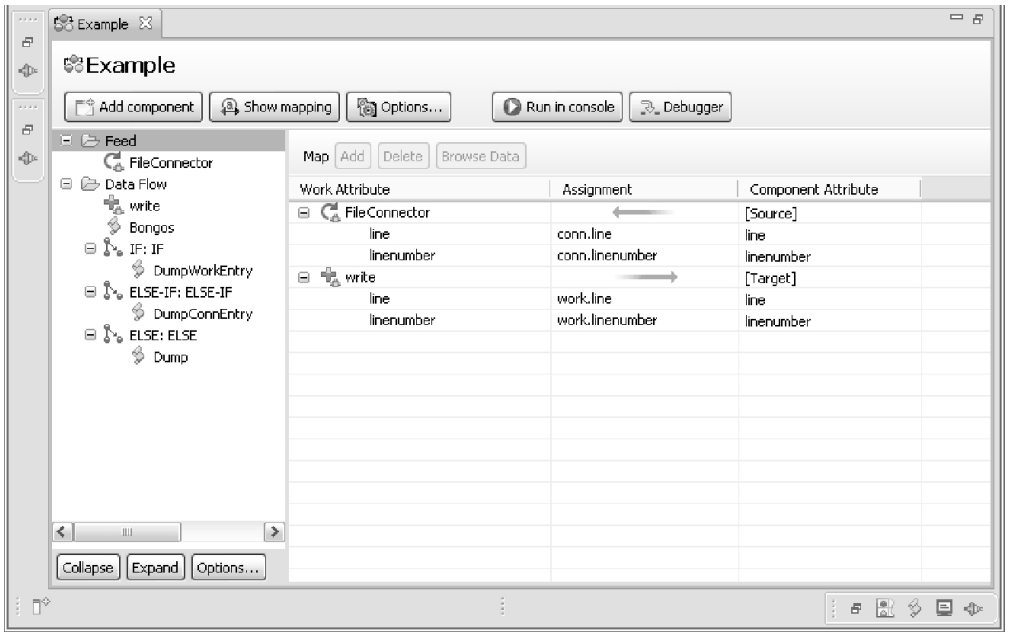


Figura 34. Mapeamento de Atributos

Nesta janela, você não vê o esquema dos componentes do AssemblyLine. Para trabalhar com o esquema, abra o editor de componentes selecionando-o na árvore à esquerda.

O cenário típico para mapeamento de atributos é primeiro descobrir o esquema do componente. Quando você cria um esquema de descoberta, a CE executará uma tarefa em segundo plano que executa o método de esquema da consulta do componente. Se nenhum esquema for retornado, a CE perguntará se você gostaria de ler uma entrada para tentar derivar o esquema a partir dela. O resultado é então preenchido novamente no esquema do componente que você está editando.

A imagem abaixo mostra o conteúdo do esquema de entrada de um componente após a descoberta de atributos. Se, por alguma razão, um componente não lhe fornecer um esquema, você poderá incluir itens de esquema manualmente utilizando o botão **Incluir...** da barra de ferramentas ou reutilizar um esquema de outra configuração de componente com a opção **Alterar Herança**.

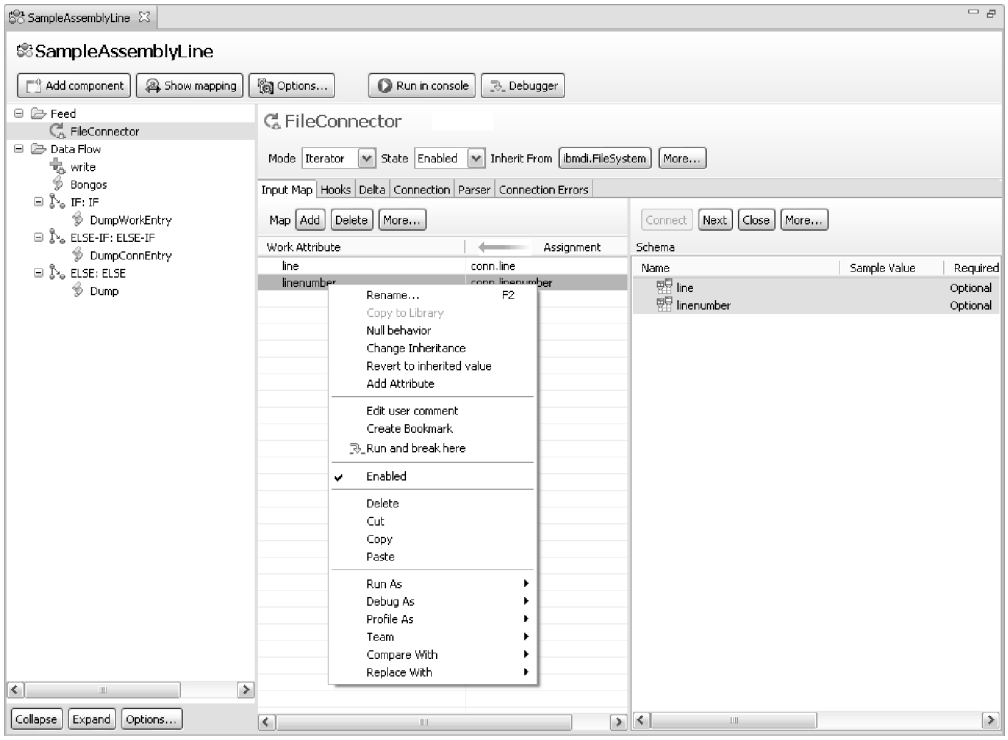


Figura 35. Mapeamento de Atributos, com Atributos Descobertos

Você pode usar também o menu suspenso da barra de títulos para alterar a herança da configuração do esquema.

Com um esquema, você pode arrastar e soltar itens individuais no mapa de atributos ou usar a função **Mapear Atributo** a partir do menu de contexto e modificar o mapeamento, se for necessário.

Figura 36. Alterando a Herança do Mapa de Atributos

Nota: A funcionalidade Arrastar e soltar depende de uma determinada extensão em seu ambiente de janelas. Em particular, nos sistemas UNIX, o Common Desktop Environment (CDE) não o fornece, portanto, para configurar o

mapeamento você precisará usar a função **Mapear Atributo** do menu de contexto.

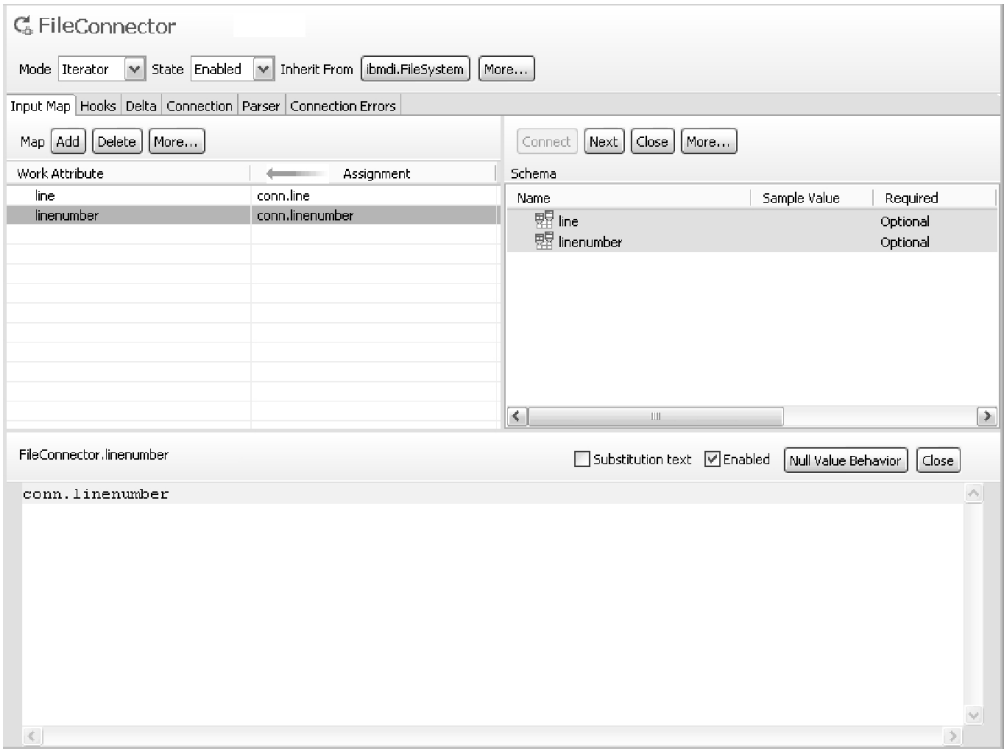


Figura 37. Mapeamento de Atributos, com a Janela de Edição JavaScript para Atributo Individual

Se você não tiver um esquema ou desejar incluir atributos independentemente do esquema, poderá criar um. Use o botão **Incluir** para incluir um novo atributo no mapa. Nomeie o atributo e uma expressão "conn.*nome-do-atributo*" ou "work.*nome-do-atributo*" será designada ao novo atributo. Isso poderá ser feito nas janelas Editor de AssemblyLine e Editor do Conector.

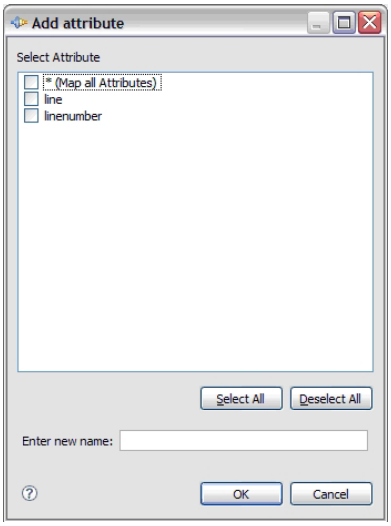


Figura 38. Diálogo Incluir Atributo

Um diálogo aparece com um campo de texto editável em que você pode digitar o nome do novo atributo. A lista acima contém todos os nomes de atributos conhecidos do esquema; você pode selecionar aqueles que deseja incluir no mapa de atributos.

À medida que você inclui mais componentes no AssemblyLine, pode arrastar atributos entre eles onde fizer sentido. Arrastar um componente em outro mapeará todos os atributos mapeados para o componente de destino. Também é possível arrastar atributos do mapa de atributos em componentes do painel esquedo, mostrando todos os componentes do AssemblyLine. Isso executará um mapa simples de todos esses itens arrastados. Isso é semelhante a arrastá-los no componente do painel de mapa de atributos.

O conceito de Mapeamento de Atributos é tratado de maneira totalmente intensiva, repleta de exemplos, no *Introdução*.

Dependendo do Conector, e do modo no qual ele é configurado, haverá diferentes guias na janela de configuração do Conector.

- Conectores em um modo que suporta entrada de um sistema conectado terão uma seção chamada **Atributos de Entrada**.
- Conectores em um modo que suporta saída para um sistema conectado terão uma seção chamada **Atributos de Saída**.
- Alguns Conectores suportam modos que podem fazer Entrada e Saída. Se configurados dessa maneira, você verá uma seção **Atributos de Entrada**, bem como uma seção **Atributos de Saída**.

Mapas de Atributo Externos

Os mapas de atributo podem herdar dos arquivos de mapeamento de atributo externo. Um arquivo de mapeamento de atributo externo é um arquivo de texto que contém itens de mapa de atributo iguais ao que você tem na tela de mapeamento real. A diferença é que o arquivo externo utiliza um formato diferente da estrutura XML interna. Isso facilita para você configurar o mapa de atributo para qualquer conector sem precisar recorrer ao CE. O CE fornece essa opção no diálogo de herança para mapas de atributo:

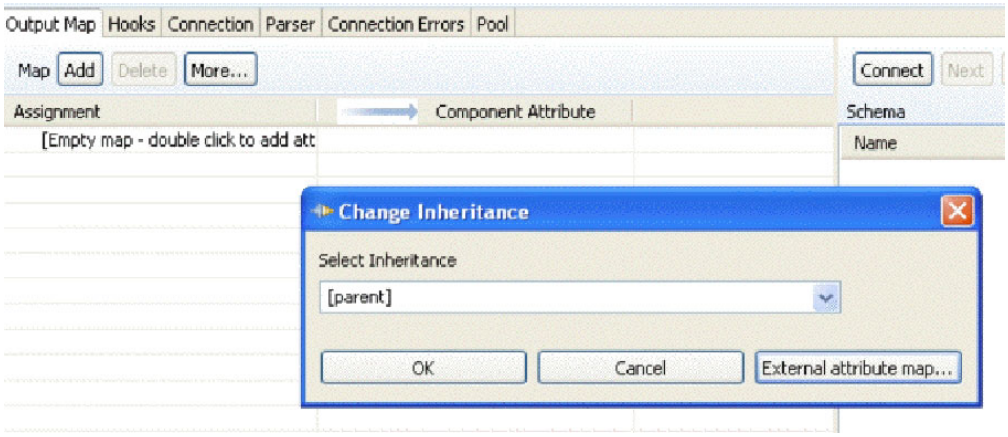


Figura 39. Mapa de Atributo: Diálogo de Herança

Clique no botão **Mapa de Atributo Externo...** para escolher um arquivo existente ou digite "`file:`" seguido pelo caminho completo para o arquivo de mapeamento

de atributo. Se quiser utilizar nomes de caminho relativo, prefixe o nome do arquivo com um ponto e uma barra (./).

Mapeamento de Atributos de Entrada:

O Mapeamento de Atributos de Entrada é o processo que executa o movimento de dados da origem de entrada para a Entrada de trabalho no AssemblyLine. Os Mapas de Atributos de Entrada são mostrados na janela Mapas de Atributos do Conector, quando abertos no Editor do Conector, com uma seta apontando para o Conector a partir de uma entidade chamada de "[Origem]". Eles são mostrados também na janela Esquema, em Mapa de Atributos de Entrada.

Antes de Iniciar

Para poder configurar o Mapa de Atributos de Entrada, o Conector deve ser configurado como um modo que suporta entrada, na barra de ferramentas do Conector. Os modos que suportam entrada são geralmente Repetidor, Consulta e Servidor.

Em seguida, na seção **Mapa de Entrada**, você seleciona esses Atributos na origem de entrada que você deseja processar no AssemblyLine.

Sobre Esta Tarefa

Os Conectores a serem configurados no Mapeamento de Atributos de Entrada podem residir na `<área de trabalho>/Recursos/Conectores`, ou em sua posição designada no AssemblyLine.

Procedimento

1. Clique em **Mapa de Entrada**.
2. Clique em **Conectar** e em **Avançar** para obter o esquema para várias origens de dados. Alguns Conectores ou combinações Conector-Analisador têm esquemas predefinidos, enquanto outros solicitam que você leia uma entrada de amostra da origem de dados e a examine para descobrir atributos.
3. Por último, selecione Atributos na lista **Esquema** e depois arraste-os no Mapa de Atributos ou inclua-os manualmente com os botões Incluir e Excluir. O Mapa de Atributos controla quais Atributos são trazidos para o AL para processamento, bem como quaisquer transformações que você especificar.

O que Fazer Depois

Esses Atributos mapeados são recuperados da origem de dados, colocados na entrada *Trabalho* e transmitidos para Conectores subsequentes da seção Fluxo do AssemblyLine.

Se você não criou o Conector diretamente em um AssemblyLine, então para usar esse Conector em um AssemblyLine, arraste o Conector de seu local em `<área de trabalho>/Recursos/Conectores` para a seção **Alimentação** de um AssemblyLine.

Mapeamento do Atributo de Saída:

Mapeamento do Atributo de Saída é o processo que executa a movimentação de dados da Entrada de trabalho no AssemblyLine para o destino de saída do sistema conectado. Os mapas do Atributo de Saída são mostrados na janela Mapas de Atributos do Conector, quando exibidos no Editor do Conector, com uma seta

apontando do Conector para uma entidade conhecida como "[Destino]". Eles são mostrados também na janela Esquema, no mapa Atributo de Saída.

Antes de Iniciar

Para poder configurar o mapa Atributo de Saída, o Conector deve estar configurado para um modo que suporta saída, na barra de ferramentas do Conector. O modo típico de saída é AddOnly. Alguns modos, como CallReply, suportam entrada e saída.

Em seguida, no **Mapa de Saída**, na seção Atributos de Saída, você seleciona esses Atributos na Entrada de trabalho do AssemblyLine que você deseja que saiam para o sistema conectado.

Sobre Esta Tarefa

Os Conectores a serem configurados no mapeamento Atributo de Saída podem residir na `<área de trabalho>/Recursos/Conectores`, ou em sua posição designada no AssemblyLine. No entanto, quando o Conector estiver na `<área de trabalho>/Recursos/Conectores` apenas, ou seja, não for um membro de um AssemblyLine, você não poderá arrastar com facilidade os atributos da Entrada de trabalho para o mapa Atributo de Saída. Nesse caso, arraste o Conector para um AssemblyLine, ou crie os mapeamentos manualmente, clicando em **Incluir** na janela Mapas de Atributos, ou então clicando com o botão direito do mouse no Conector da janela Mapas de Atributos e selecione **Incluir Item do Mapa de Atributos**.

Procedimento

1. Clique em **Mapa de Saída**.
2. Clique em **Conectar** para obter o esquema para a origem de dados. Alguns Conectores ou combinações de Conector-Analisador têm esquemas predefinidos, que serão exibidos. Muitos Conectores, no entanto, não têm.
3. Se o seu Conector estiver em um AssemblyLine, arraste os atributos da Entrada de trabalho mapeados anteriormente no Conector da janela Mapas de Atributos do editor do AssemblyLine. Ou então, crie Atributos manualmente - a correspondência de nomes ocorre no tempo de execução. Por exemplo, um item do mapa Atributo do Mapa de Saída criado como `some_attribute` faz com que um atributo de Entrada de trabalho chamado some_attribute seja mapeado para um atributo do sistema conectado do mesmo nome.

O que Fazer Depois

Esses Atributos mapeados são recuperados da Entrada de *Trabalho* quando esse Conector é chamado no Fluxo do AssemblyLine e saem para o sistema conectado.

Se você não criou o Conector diretamente em um AssemblyLine, então para usar esse Conector em um AssemblyLine, arraste o Conector de seu local na `<área de trabalho>/Recursos/Conectores` para a seção **Fluxo** de um AssemblyLine.

O Editor de Conector

O editor de conector é usado ao editar arquivos de conector ou ao usar a função **Editar** em um Conector no AssemblyLine.

A criação de um Conector é descrita na seção “Criando um Conector” na página 116.

O editor usa os mesmos widgets que você localiza nos assistentes e nos diálogos pop-up para conectores. O editor consiste em seis guias, nas quais os painéis de configuração para vários aspectos do conector são mostrados. Na parte superior, estão os atributos principais do conector, como seu modo, estado e outras opções gerais do conector.

As guias são:

1. "Mapas de Atributos de Entrada e de Saída" na página 117
2. "Ganchos" na página 117
3. "Conexão" na página 118
4. "Analisador" na página 119
5. "Critérios de Link" na página 119
6. "Erros de Conexão" na página 121
7. "Delta" na página 123
8. "Conjunto" na página 125
9. "Herança do Conector" na página 125

Criando um Conector

Criar um Conector envolve decidir onde o Conector residirá e quais os parâmetros iniciais para designá-lo.

Antes de Iniciar

Decida onde o Conector deve residir; há dois locais:

1. Conectores destinados à reutilização e compartilhamento de recursos residem no diretório `<workspace>/Resources/Connectors`. Esse é geralmente o melhor local para criar e manter seus Conectores. Conectores definidos dessa maneira são incluídos nos AssemblyLines arrastando-os para o local apropriado.

   Depois disso, você pode alterar essas poucas definições do Conector, tal como executar sua função designada no AssemblyLine, mas manter a maioria das configurações herdadas e, portanto, inalteradas a partir das definições da seção Recursos.
2. Você pode também criar um Conector diretamente em um AssemblyLine; os Conectores definidos dessa maneira são definições ad hoc que são válidas apenas no contexto desse AssemblyLine específico.

Sobre Esta Tarefa

Os Conectores formam a estrutura de qualquer solução criada com o IBM Security Directory Integrator; eles estabelecem a conexão com os sistemas com os quais você deseja trocar dados.

Procedimento

1. Clique com o botão direito do mouse e selecione **Recursos** > **Conectores** > **Novo Conector...** em sua área de trabalho ou selecione **Arquivo** > **Novo** > **Conector**
2. Navegue até o local em que deseja manter seu novo Conector e nomeie seu novo Conector.
   a. O local recomendado é `<workspace>/Resources/Connectors`.

b. Ou então, você pode criar o novo Conector diretamente em seu AssemblyLine. Navegue para o local do AssemblyLine de destino; a seção Alimentação para Conectores no modo Agente Iterativo e Servidor ou Fluxo para todos os outros modos.

3. Clique em **Concluir** para criar o Conector.
4. Na guia **Conexão**, defina o modo do seu novo Conector para o modo desejado.
5. Defina os parâmetros de conexão desse Conector na guia **Conexão**; os parâmetros obrigatórios são marcados com um asterisco (*). Alguns Conectores requerem que você configure um Analisador como na sub-guia **Analisador**.
6. Vá para a janela **Mapa de Atributos** na janela de configuração Conector para descobrir ou definir o esquema para essa origem de dados: clique em **Descobrir Atributos** para obter o esquema para a origem de dados. Alguns Conectores ou combinações Conector-Analisador têm esquemas predefinidos. Aqueles que não têm, solicitam que você leia uma entrada de amostra da origem de dados e examine-a para descobrir atributos.

O que Fazer Depois

Uma vez que o Conector foi definido, você está pronto para configurar as partes de informações conhecidas como *Atributos* que fluirão de e para o AssemblyLine. Esse processo é chamado de *Mapeamento de Atributos* e o ponto de vista é a partir do AssemblyLine. Por essa razão, a definição do Mapeamento de atributos a partir do sistema conectado, por meio do Conector de entrada até a Entrada de trabalho no AssemblyLine é feita no *Mapa de Atributos de Entrada*; e o inverso, o Mapeamento de atributos a partir da Entrada de trabalho, por meio do Conector de saída até o sistema conectado é feito no *Mapa de Atributos de Saída*.

Mapas de Atributos de Entrada e de Saída

A guia Mapa de Atributos mostra os mapas de atributos de entrada e de saída e esquemas para um componente.

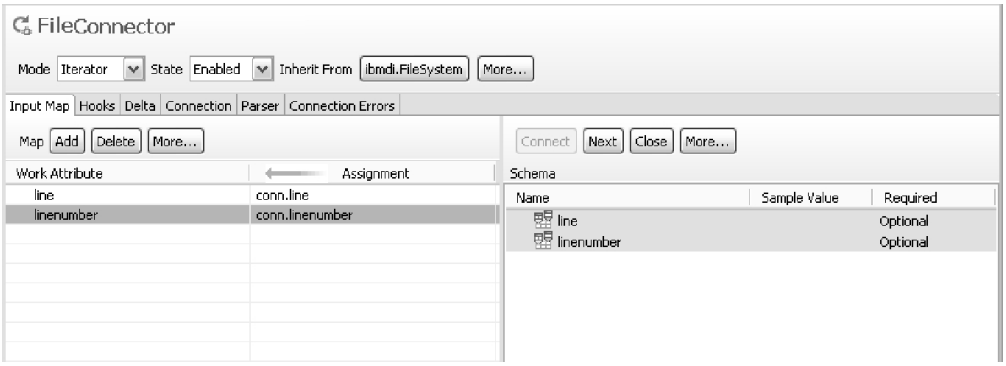


Figura 40. Janela Mapa de Atributos

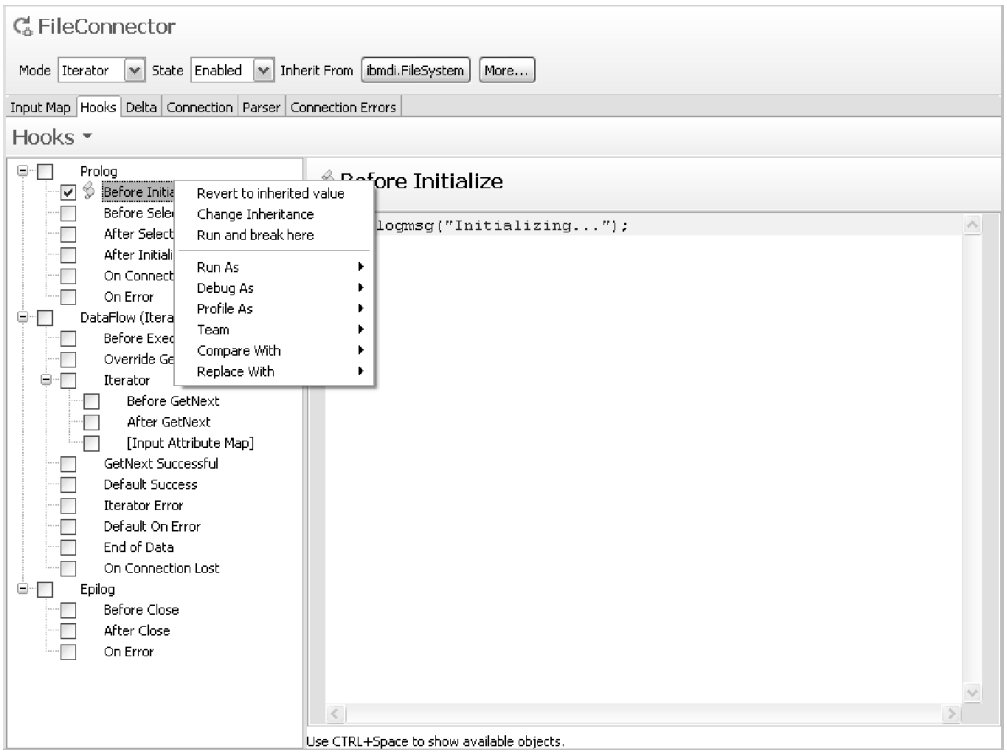
Consulte a seção "Mapeamento de Atributos e Esquema" na página 109 do editor do AssemblyLine para obter uma descrição desta janela.

Ganchos

A guia Ganchos mostra todos os ganchos do conector.

Use a caixa de opção para ativar/desativar um gancho e selecione o gancho para editar seu conteúdo. Quando você modifica o conteúdo de um gancho, ele é ativado automaticamente. Os ganchos que contêm código de script terão um ícone

de script na visualização em árvore para que você possa identificar com rapidez se um gancho tem conteúdo ou não. Observe que se um gancho estiver ativado, ele será executado quando for atingido no fluxo de execução, quer contenha algum script ou não.



Consulte a seção "Fluxo do AssemblyLine e Ganchos" na página 33 para obter uma discussão dos diversos ganchos, no nível do AssemblyLine e no nível do componente individual.

Conexão

Figura 41. Guia Conexão

Os parâmetros da guia Conexão são altamente específicos para o componente que você deseja configurar. Consulte a especificação individual do componente no *Referência*.

Analisador

Se um conector puder usar (ou exigir) um analisador, você terá também uma guia para ele.

Use o botão da barra de ferramentas **Selecionar Analisador** para alterar o analisador do conector.

Por exemplo, o Analisador do Leitor de Linhas possui uma tela de configuração como a mostrada abaixo:

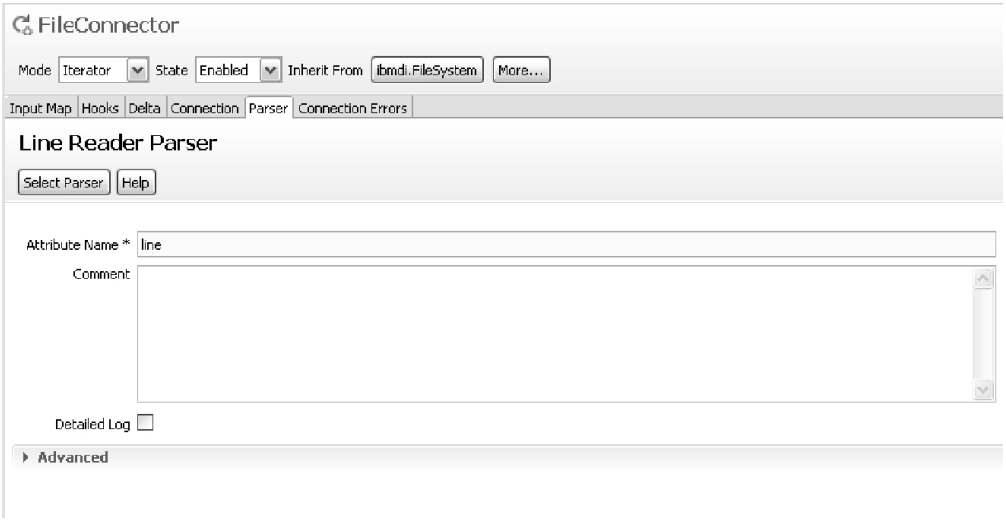


Figura 42. Analisador do Leitor de Linhas

Critérios de Link

Quando um componente exige critérios de vinculação, você verá a guia Critérios de Link.

Se um componente realmente exibir uma guia Critérios de Link dependerá não apenas do tipo de componente, mas também do seu modo. Consulte "Critérios de Link" na página 20 para obter informações adicionais.

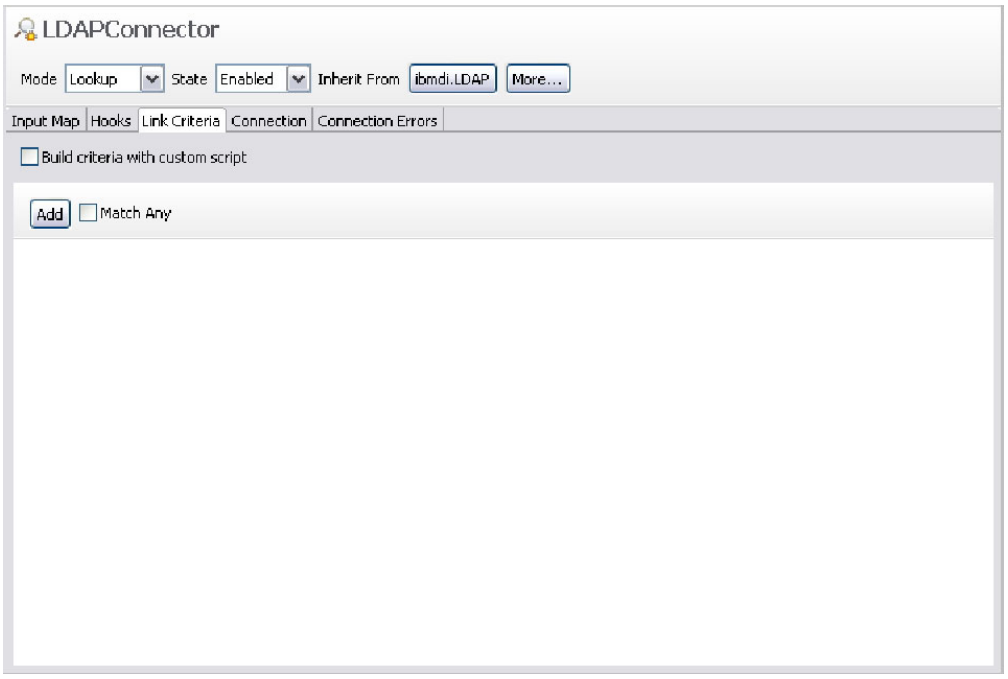


Figura 43. Guia Critérios de Link

Use o botão **Incluir** para incluir uma nova linha de controle da visualização. Use o botão **Excluir** para remover linhas individuais. Na visualização, você especifica o nome do atributo, o operando (por exemplo, igual, contém) e o valor de correspondência. O valor de correspondência pode ser uma constante ou uma expressão. Use o botão  para abrir o editor de expressão:

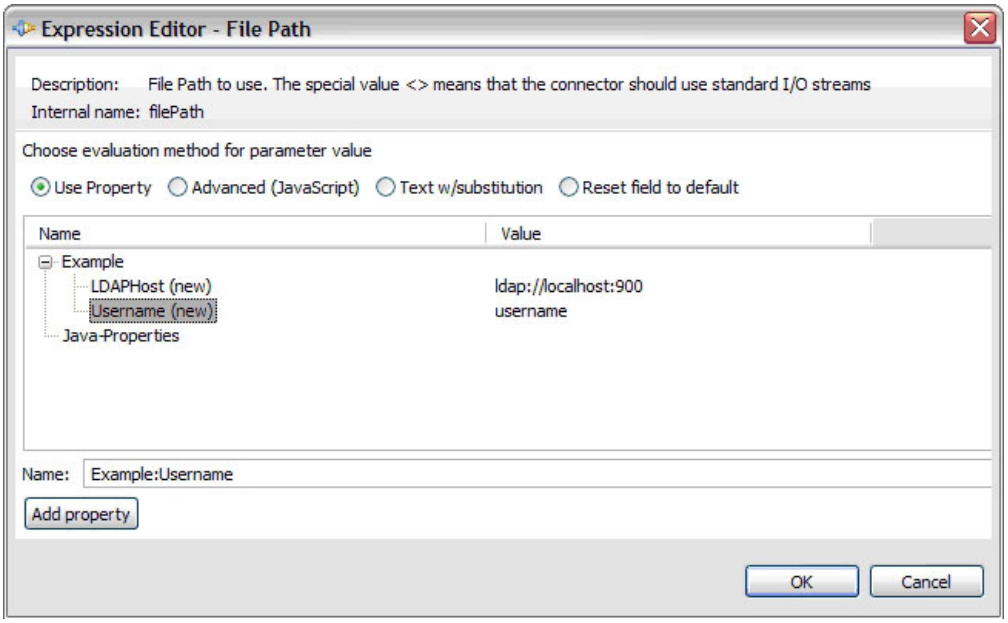


Figura 44. Janela Editor de Expressão, Modo Simples

No editor de expressão, você pode escolher de maneira conveniente uma propriedade de um de seus arquivos de propriedades ou usar o modo avançado em que você retorna um valor baseado em código JavaScript. Marque a caixa de

opção **Avançado (JavaScript)** para se alternar entre a seleção de propriedade e o código JavaScript. Você pode usar apenas um ou outro.

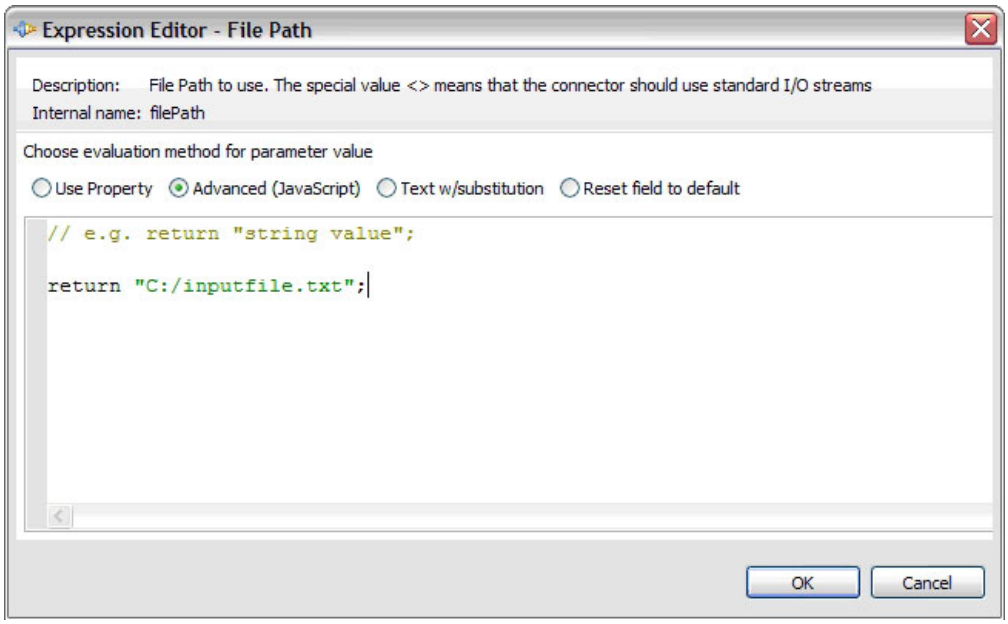
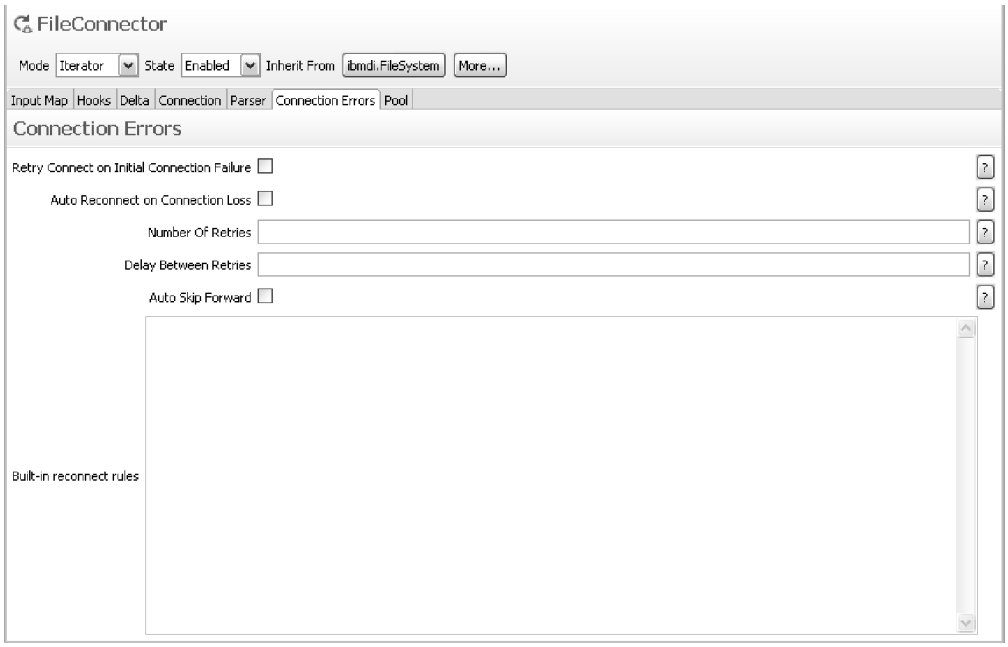


Figura 45. Janela Editor de Expressão, Modo Avançado (JavaScript)

Erros de Conexão

A guia Erros de Conexão é onde você configura a resiliência da conexão, ou seja, o comportamento automatizado a ser executado quando a conexão estabelecida pelo seu componente falhar.

Opcionalmente, quando você seleciona **Tentar Conexão Novamente na Falha da Conexão Inicial**, você pode configurar se sua tentativa de conexão emitirá uma exceção ao falhar durante a inicialização ou se tentará novamente. Se essa sinalização estiver configurada e uma conexão não puder ser estabelecida quando o conector estiver sendo inicializado, uma tentativa de "reconexão" será feita; não é realmente uma reconexão, uma vez que uma conexão não foi estabelecida na primeira vez, mas geralmente o mesmo mecanismo das situações que podem ocorrer quando uma conexão estabelecida for perdida.



Para conexões estabelecidas, os parâmetros restantes têm a seguinte significância:

Reconexão Automática para Perda de Conexão
Se esse sinalizador estiver configurado e a conexão for perdida depois que o conector for inicializado, uma tentativa de reconexão será feita.

Número de Tentativas
O número de vezes que uma tentativa de reconexão será feita quando ocorrer um problema, antes de desistir. Se ocorrer um novo problema posteriormente, o mesmo número de tentativas será feito.

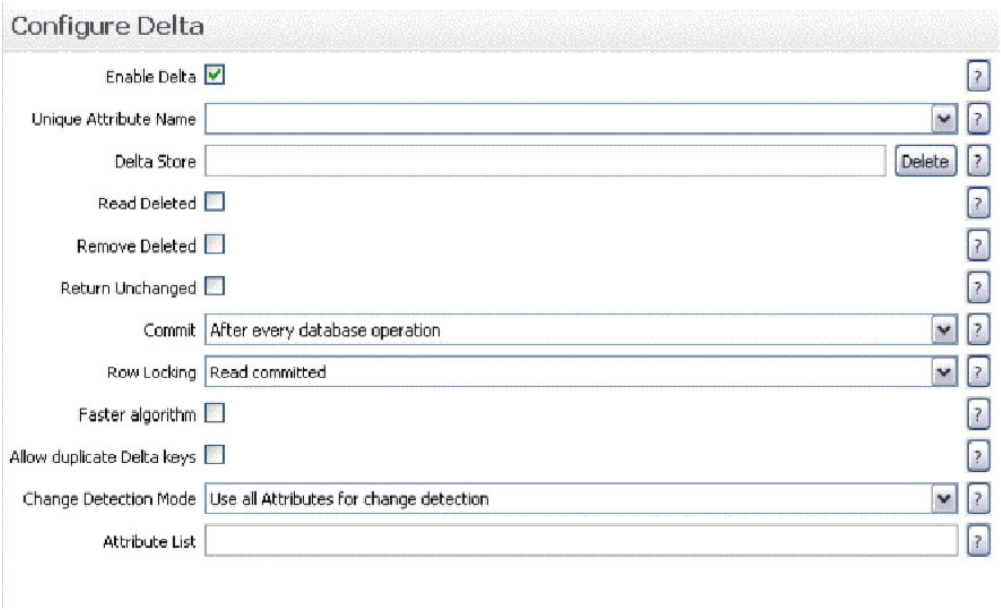
Atraso Entre Novas Tentativas
O número de segundos a aguardar entre cada tentativa de reconexão e antes da primeira tentativa de reconexão.

Ignorar Redirecionamento Automaticamente
Após uma reconexão, ignore automaticamente o redirecionamento quantas vezes o número de leituras bem-sucedidas.

Regras de reconexão internas
Elas são ligadas ao Mecanismo de Regras de Reconexão; consulte a seção correspondente em *Instalando e administrando*

para obter informações adicionais.

Delta



Essa guia está disponível apenas no modo Repetidor.

Os parâmetros dessa guia têm a seguinte significância:

Ativar Delta
Essa é a opção principal do mecanismo Delta deste Conector. Se não estiver selecionada, os parâmetros abaixo não estarão ativados.

Nome de Atributo Exclusivo
Esse é o nome de um atributo, ou uma opção de vários atributos de entrada separados por um "+", que contém um valor exclusivo em uma determinada origem de dados. As origens de dados com chaves duplicadas não podem ficar sujeitas à função delta, exceto quando **Permitir chaves Delta duplicadas** estiver ativada. Consulte “Detecção Delta” na página 16 para obter informações adicionais.

Armazenamento Delta
A tabela do Armazenamento do Sistema que contém as informações Delta de execuções anteriores deste Conector, para que seja capaz de detectar diferenças em execuções subsequentes.

Ler Excluídas
Se marcada, o AssemblyLine injetará entradas excluídas na execução do AssemblyLine quando o Repetidor tiver concluído a iteração, ou seja, terminado a entrada. O código de operação indicará que essa entrada foi excluída da origem de entrada. Observe que Entradas marcadas como excluídas não são removidas do Armazenamento Delta, a menos que você ative também o sinalizador Remover Excluídas.

Remover Excluídas
Se marcada, as entradas excluídas da origem de entrada serão excluídas do Armazenamento Delta, de modo que não sejam detectadas novamente em execuções subsequentes.

Retornar Não Alteradas
Se marcada, todas as entradas não alteradas desta execução são injetadas no AssemblyLine.

Consolidar
Selecione quando consolidar alterações no Armazenamento Delta como resultado de iteração por meio da entrada. As opções são:

- Depois de cada operação do banco de dados
- No final do Ciclo do AL
- No fechamento do Conector
- Nenhuma confirmação automática

O padrão é **Após cada operação do banco de dados**.

Bloqueio de Linha
Seleciona o nível de isolamento de transação para a conexão com o Armazenamento Delta. Esse parâmetro aborda a necessidade de um bloqueio de linha em uma tabela de Armazenamento Delta quando vários Clientes do BD acessam os mesmos dados através da configuração de um *Nível de Isolamento de Transação*. A configuração de um nível de isolamento maior reduz anomalias da transação conhecidas como 'leituras sujas', 'leituras não repetíveis' e 'leituras fantasma' usando bloqueios de linha e de tabela

As opções são:

- READ_UNCOMMITTED
- READ_COMMITTED
- REPEATABLE_READ
- SERIALIZABLE

O padrão é READ_COMMITTED.Para mais informações, consulte a seção “Bloqueio de Linha” na página 215.

Lista de Atributos
Essa é uma lista de atributos separados por vírgulas cujas mudanças serão detectadas ou ignoradas durante o processo de mudança de cálculo. As mudanças nos atributos listados serão afetadas pelo parâmetro **Modo de Detecção de Alterações**, que especifica se você deve ignorá-las ou detectá-las. Para obter mais informações sobre esse parâmetro e o próximo, consulte a seção “Detectar ou Ignorar Mudanças Apenas em Atributos Específicos” na página 216.

Modo de Detecção de Alterações
Especifique se você quer detectar ou ignorar mudanças nos Atributos listados no parâmetro **Lista de Atributos**. Os valores possíveis são:

- IGNORE_ATTRIBUTES
- DETECT_ATTRIBUTES
- DETECT_ALL

Algoritmo Mais Rápido
Quando marcada, instrui o AssemblyLine a usar um algoritmo mais rápido para computar alterações, à custa de mais uso de memória.

Permitir chaves Delta duplicadas
Quando o recurso Delta é ativado para Log de Mudanças/Conector de Detecção de Mudanças em AssemblyLines de longa execução, é possível que uma Entrada possa ser modificada mais de uma vez. Essas modificações resultarão no recebimento da Entrada uma segunda vez, o que fará com que uma exceção de chave delta Duplicada seja emitida. A verificação desse parâmetro permite que Entradas com atributos de chave

duplicada (especificadas no parâmetro **Nome do Atributo Exclusivo**) sejam processadas por Conectores do Agente Iterativo com Delta ativado.

Conjunto

A definição do conjunto de conectores de um conector é visível apenas quando o conector reside na biblioteca de conectores (ou seja, um arquivo em Projeto/Recursos/Conectores). Conectores que são abertos no AssemblyLine não terão essa guia.

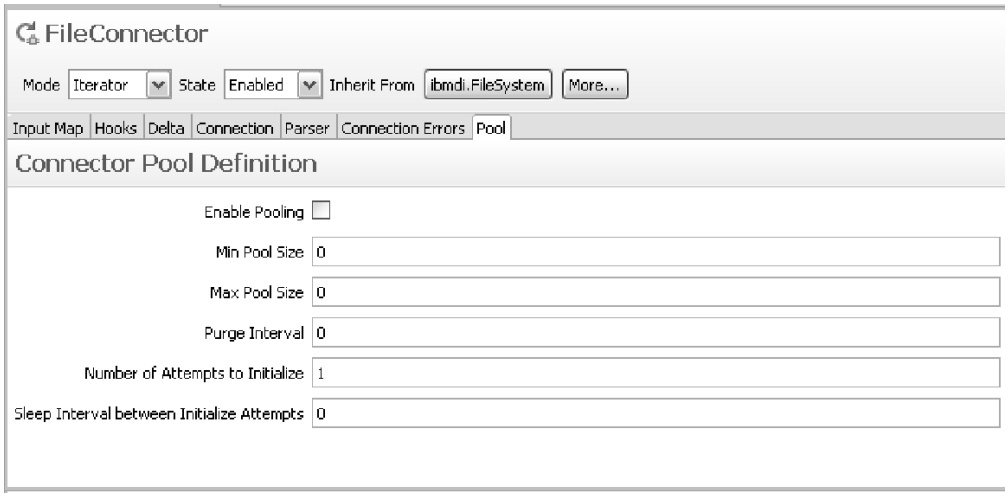


Figura 46. Guia Conjunto: Definição do Conjunto de Conectores

Quando um conector em conjunto for usado em um AssemblyLine, ele mostrará a seguinte guia:

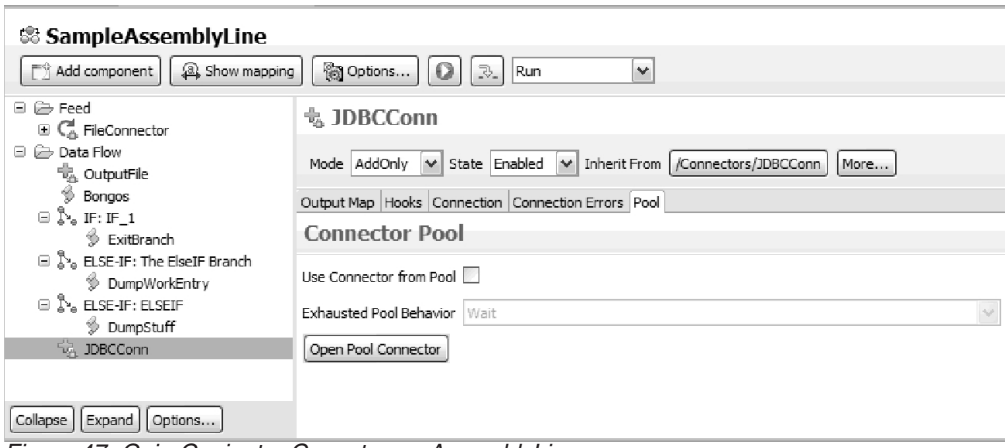


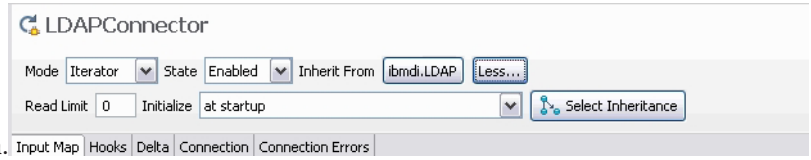
Figura 47. Guia Conjunto: Conector no AssemblyLine

Use o botão **Abrir Conector do Conjunto** para abrir o conector do conjunto.

Herança do Conector

Embora você possa alterar a herança em vários locais do editor, pode também usar o botão de herança do conector para obter uma lista completa das configurações de herança.

Use o botão **Mais...** para expandir o cabeçalho do editor do conector e clique no botão **Herança**.



O botão **Herança** abrirá um diálogo que mostra todas as configurações da herança para o conector:

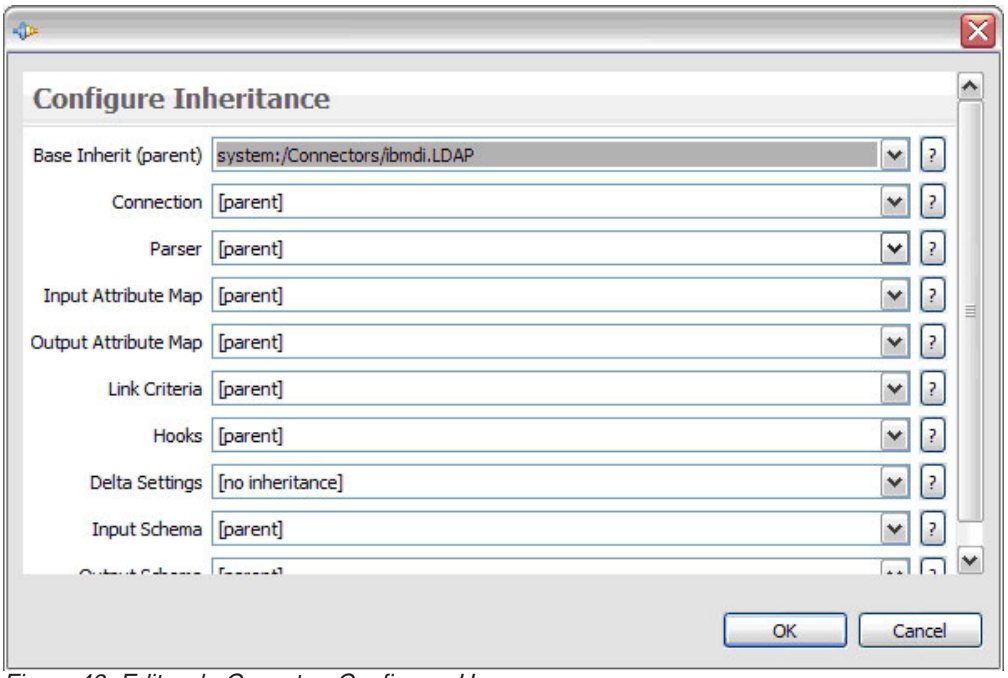


Figura 48. Editor do Conector: Configurar Herança

A notação [pai] significa que o item é herdado do componente listado no campo **Herança Base (pai)**.

Editor do Servidor

O editor do servidor é onde você define como atingir um servidor IBM Security Directory Integrator.

Um servidor é sempre definido pelo CE e é denominado “Padrão”. Na visualização dos servidores IBM Security Directory Integrator, você pode incluir novos servidores em seu projeto.

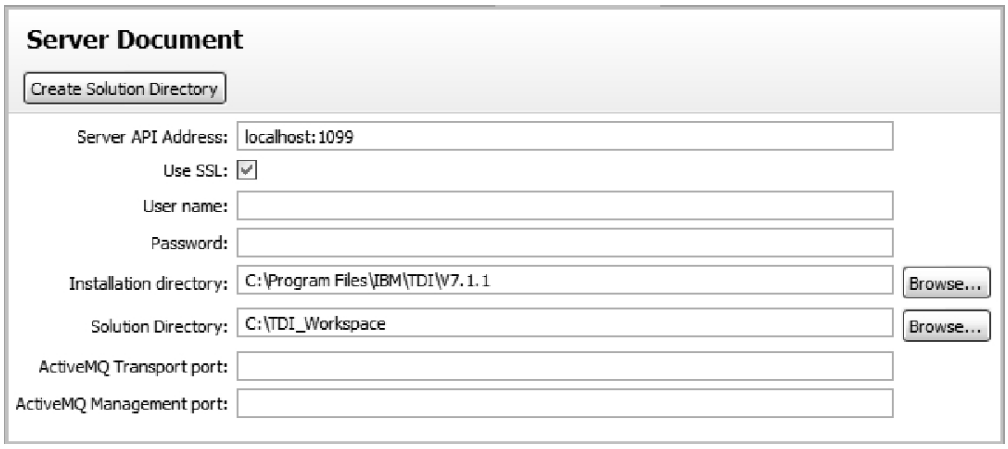


Figura 49. Editor de Documento do Servidor

O endereço da API do servidor é o endereço `host:port` do servidor IBM Security Directory Integrator. Se você especificar o diretório de instalação, o CE poderá iniciar o servidor. Antes de iniciar um novo servidor IBM Security Directory Integrator, configure todos os parâmetros e use o botão **Criar Diretório de Soluções** para criar os arquivos necessários para o servidor. Também é possível especificar portas exclusivas para o transporte e o gerenciamento do Apache ActiveMQ.

Editor de Esquema

O editor de esquema gerencia os arquivos de esquema de design.

Esses arquivos podem ser utilizados por mapas de entrada e de saída em outros editores. O esquema é apenas tempo de design (ou seja, apenas no CE) e normalmente é utilizado quando você tem esquemas grandes que não quer que apareçam no arquivo de configuração do runtime.

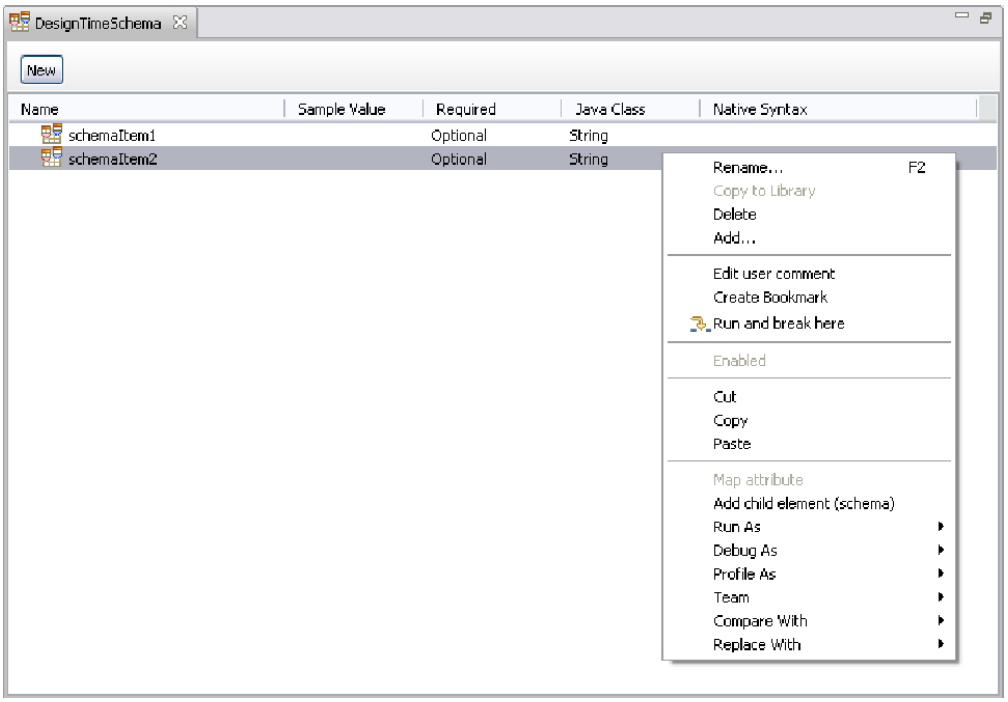


Figura 50. Editor de Esquema

O editor fornece um botão **Novo** para incluir um novo item de esquema de nível superior e um menu de contexto para operar em itens de esquema existentes.

Dados do Navegador

Os Dados do Navegador fornecem uma consulta a fundo em um sistema de destino. Atualmente, há apenas os conectores LDAP e JDBC que fornecem detalhes extra para um conector. O navegador de dados é aberto clicando com o botão direito do mouse em um Conector na biblioteca ou em um AssemblyLine.

No navegador, você pode clicar com o botão direito do mouse e escolher **Dados do Navegador** para abrir uma nova janela do editor em que você pode navegar pelos dados dentro da configuração de conexão atual pelo conector.

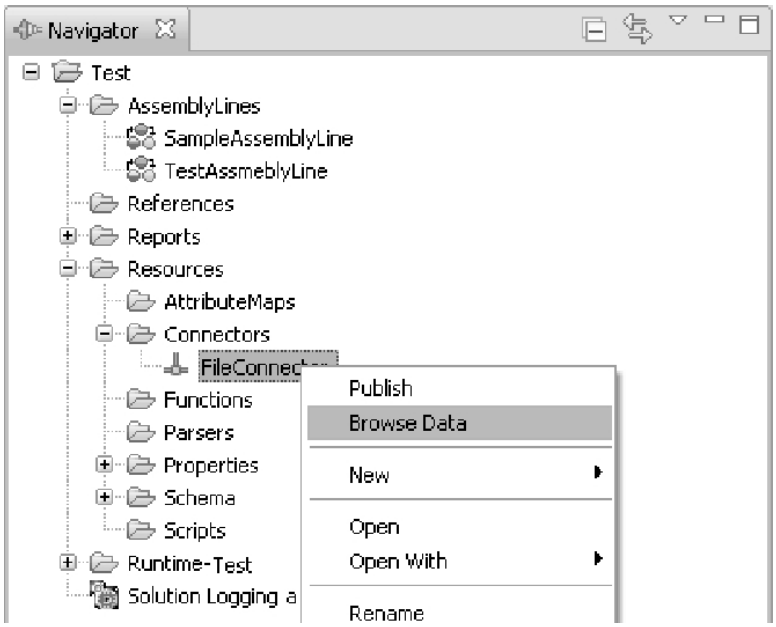


Figura 51. Dados do Navegador

No AssemblyLine, você pode fazer o mesmo e ter uma nova janela aberta para os dados do navegador:

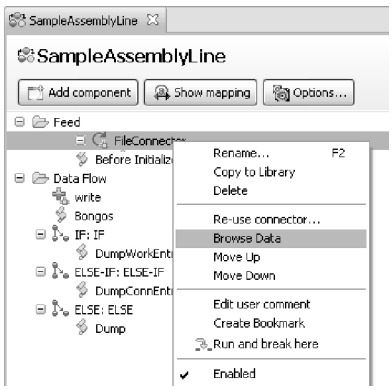


Figura 52. Dados do Navegador

Navegador de Dados Genérico

Este é o navegador de dados usado para aqueles conectores dos quais o Editor de Configuração não tem conhecimento explícito. Ele fornece uma maneira simples de navegar por um conjunto de resultados a partir da origem de dados.

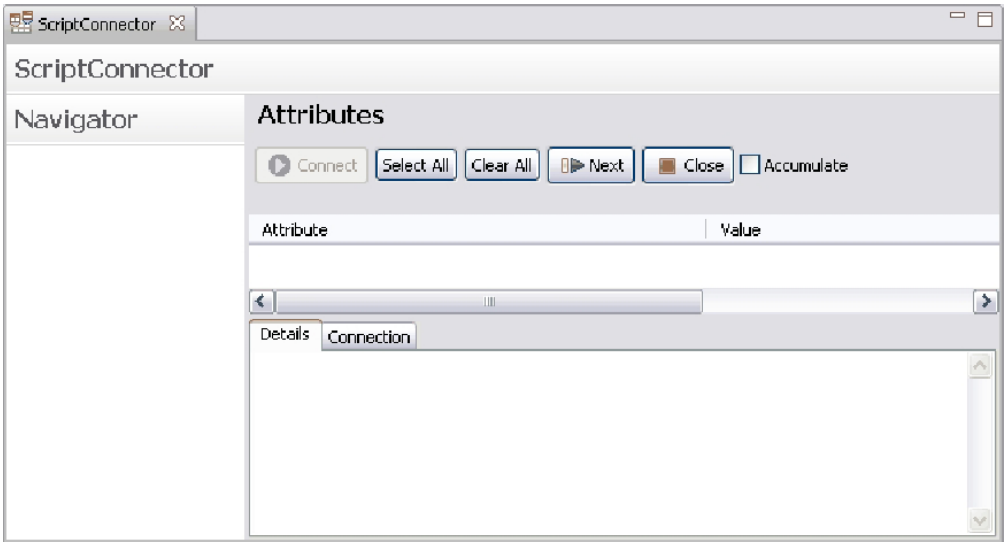


Figura 53. Navegador de Dados Genérico

A parte superior mostra uma lista de atributos lidos a partir do conector e de uma barra de ferramentas. Os atributos da lista possuem caixas de opção; quando você marca um atributo, um mapeamento de atributos é criado para esse atributo usando uma expressão simples `conn.attributename -> work.attributename`. Quando você desmarca um atributo, o mapeamento de atributos desse atributo é removido.

A barra de ferramentas tem as seguintes funções:

Tabela 8. Barra de Ferramentas Navegador de Dados

Comutar Todos	Esse comando comutará as caixas de opções de cada atributo da lista. Isso causará uma modificação no mapa de atributos de todos os atributos listados.
Acumular	Esse comando alterna se a lista de atributos descobertos será acumulada ou não. Quando você acumula atributos, cada registro lido do conector é mesclado com a lista de atributos existente. Quando você não acumula, todos os atributos são removidos antes de o próximo registro ser mostrado na lista.
	Clique nesse botão para fechar a conexão. Quando você fecha a janela do editor do navegador, a conexão é automaticamente fechada. Você geralmente desejará fechar a conexão antes de ler o próximo registro se tiver modificado as configurações de conexão desse editor.
	Clique nesse botão para ler o próximo registro do conector. Quando não houver mais registros retornados do conector, uma mensagem será mostrada à esquerda da barra de ferramentas para indicar que não há mais entradas do conector. Pressionar esse botão novamente após essa condição fará com que o conector comece a ler do começo de seu conjunto de resultados.

A parte inferior da tela mostra duas guias. A primeira guia é a guia **Detalhes** que contém detalhes sobre a seleção atual. Para o navegador de dados genérico, essa guia estará sempre vazia.

A segunda guia é a guia **Conexão**. Essa guia mostra a configuração de conexão do conector. Você pode modificar os parâmetros de conexão e salvá-los como faz normalmente quando abre o editor do Conector.

Navegador de Dados de Fluxo

O navegador de dados baseado em fluxo é usado quando um conector usa um analisador. O navegador de dados baseado em fluxo primeiro inicializará o conector e tentará obter o fluxo de entrada e mostrar os primeiros 20 K de dados de entrada na guia de detalhes.

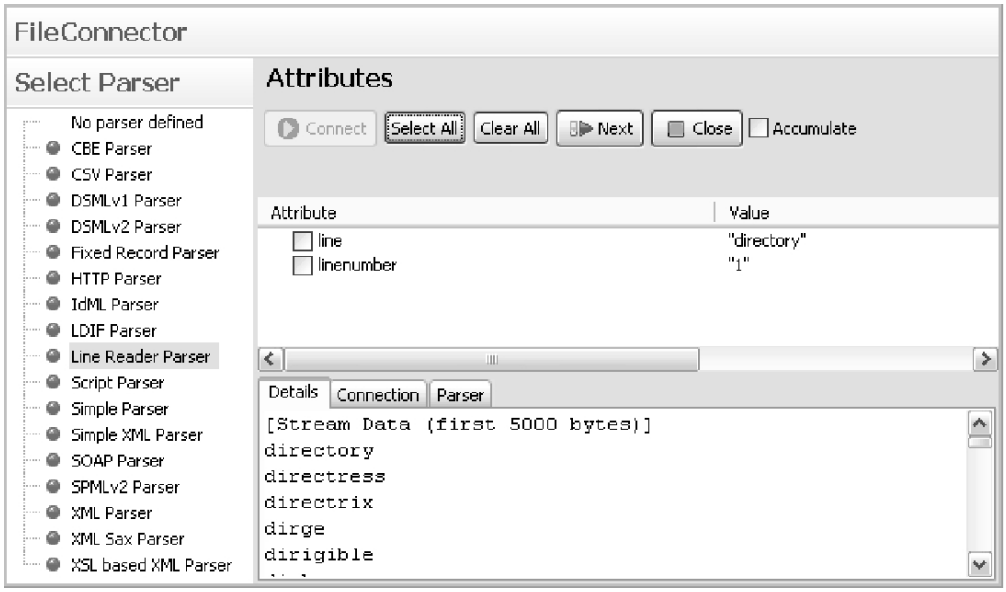


Figura 54. Navegador de Dados de Fluxo

O lado esquerdo contém agora uma lista **Selecionar Analisador**, na qual você pode selecionar um analisador, a fim de tentar ler o conteúdo do fluxo de entrada e ver se ele corresponde às suas expectativas. Ao selecionar um analisador, a guia **Analisador** será atualizada com o formulário de configuração para esse analisador. Sempre que o analisador for alterado o conector será fechado para que você possa ver facilmente se um analisador está apto a interpretar a entrada, selecionando continuamente um analisador seguido por um ler-próximo.

Nota: Ao selecionar um analisador na tabela, você também modifica a configuração do conector para usar esse analisador. Ao fechar e salvar (ou usar a função **Arquivo** > **Salvar**) você atualiza efetivamente a configuração com os parâmetros configurados atualmente.

Navegador de Dados JDBC

O navegador de dados JDBC mostra todas as tabelas e visualizações do lado esquerdo. A guia de detalhes mostra informações sobre o item selecionado nesta lista.

Figura 55. Navegador de Dados JDBC

A guia **Detalhes** mostra as informações do sistema obtidas a partir do objeto de conexão JDBC. A visualização em árvore à esquerda mostra todas as tabelas e visualizações com suas colunas como entradas filhas. Quando você seleciona uma tabela ou visualização, a guia de detalhes será preenchida com a sintaxe da tabela inteira. Por exemplo, se você selecionar a tabela "IDI_PS_DEFAULT", verá algo semelhante a isto:

Figura 56. Detalhes da Tabela JDBC

Quando você seleciona uma coluna em uma tabela ou visualização, verá os detalhes apenas dessa coluna.

Figura 57. Opção "Usar como Tabela"

Você pode também clicar com o botão direito do mouse no nome de uma tabela e usar a função **Usar como tabela** para atualizar o parâmetro **`Nome da Tabela`** da configuração do conector JDBC.

O botão  da barra de tarefas do cabeçalho de **Tabelas JDBC** faz uma redescoberta da conexão. Você precisa usá-la apenas se a descoberta inicial falhar ou se você tiver alterado a URL do JDBC na guia de conexão.

Navegador de Dados LDAP

O navegador de dados LDAP mostra o esquema e os prefixos de contexto (bases de pesquisa) que o servidor LDAP fornece.

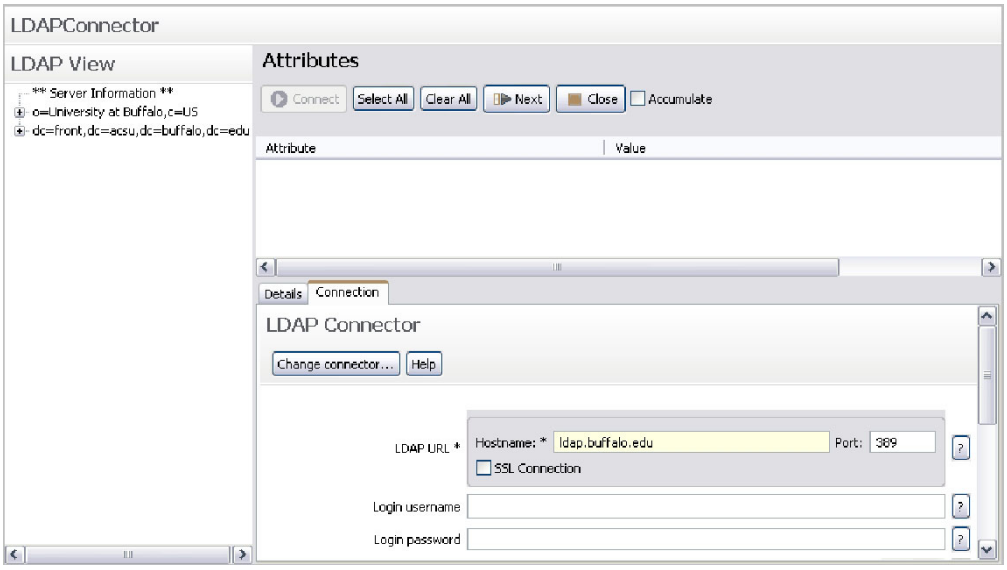


Figura 58. Navegador de Dados LDAP

A Visualização LDAP contém informações do servidor e bases de pesquisa que o servidor LDAP fornece. Baseando sua seleção nesta árvore, você verá resultados diferentes. Se você selecionar um dos nós sem esquema, deverá ver um dump

detalhado dessa entrada específica na guia de detalhes, conforme mostrado abaixo:

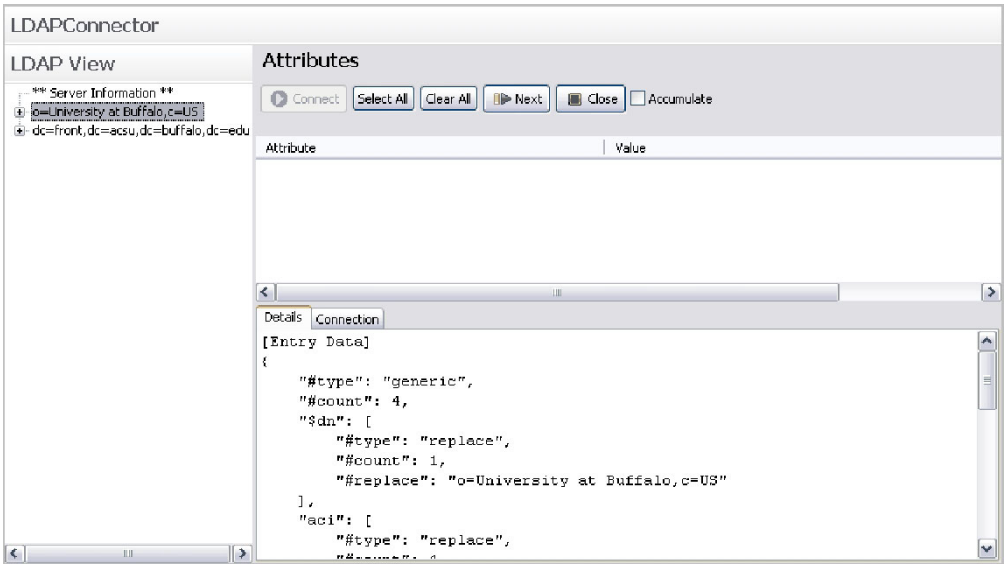


Figura 59. Entrada do Navegador de Dados LDAP

Quando você seleciona um item do esquema, vê os detalhes na guia de detalhes, bem como tem sua lista de atributos atualizada com informações provenientes do item do esquema. Isso é muito útil se você estiver lendo ou gravando um esquema específico:

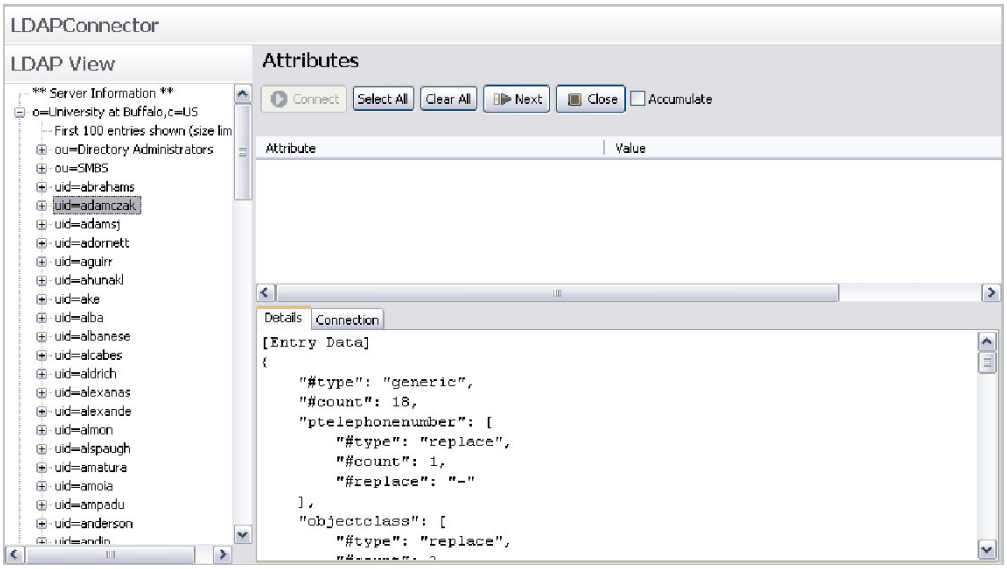


Figura 60. Item do Esquema do Navegador de Dados LDAP

Escolher uma classe de objeto no nó do esquema permite criar rapidamente um mapeamento de atributos para essa classe. A coluna de valor contém agora informações sobre o atributo. O valor “MAY” significa que ele é opcional, considerando que “MUST” significa que ele é obrigatório (ao incluir entradas). O valor entre parênteses mostra a classe de objeto que define o atributo; as classes de objeto LDAP são hierárquicas.

Você pode também atualizar rapidamente o parâmetro **Base de Pesquisa** do conector LDAP, escolhendo **Usar como base de pesquisa** no menu de contexto.

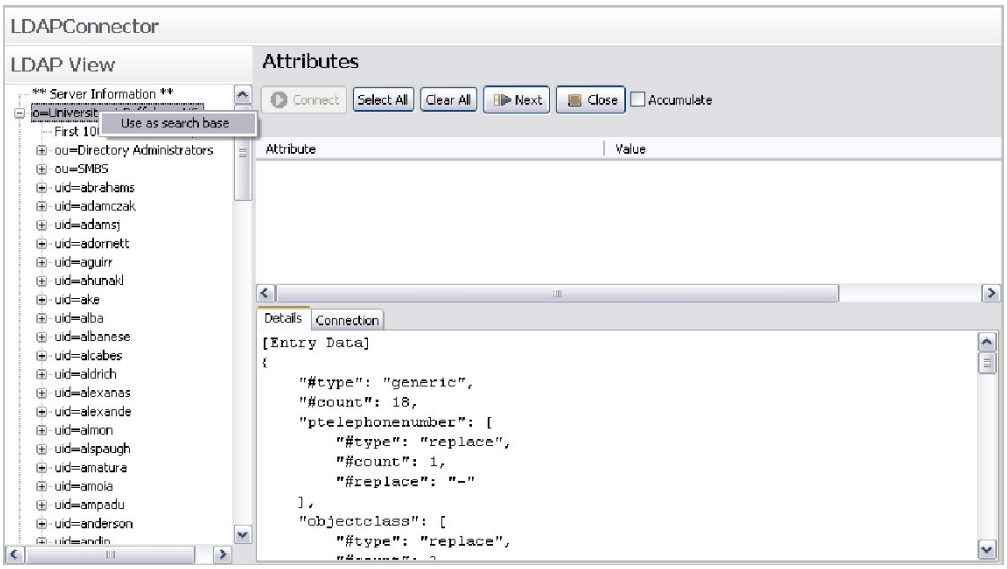


Figura 61. Opção do Menu de Contexto "Usar Como Base de Pesquisa"

Editor de Formulários

O editor de formulários é usado para customizar o formulário de parâmetros da conexão de um componente. Isso pode ser aplicado apenas a componentes da pasta Recursos (exceto arquivos de propriedades).

Para customizar um formulário, você deve abrir o componente com o *Editor de Formulários*.

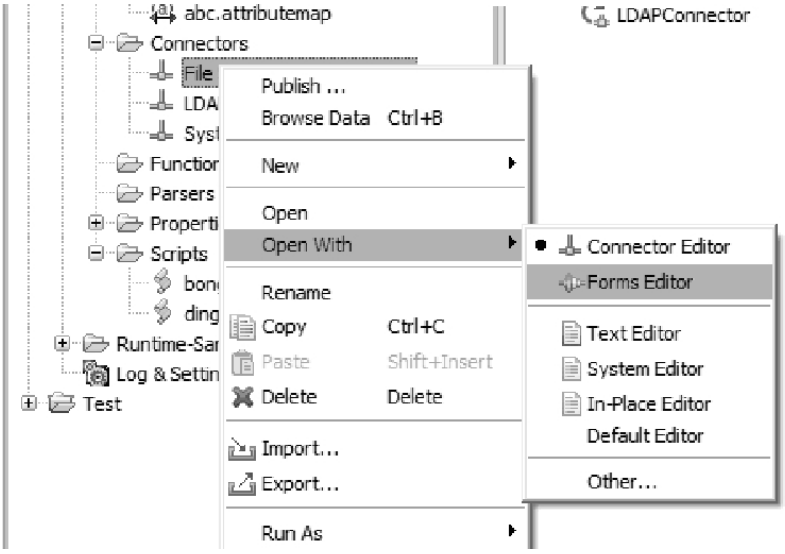
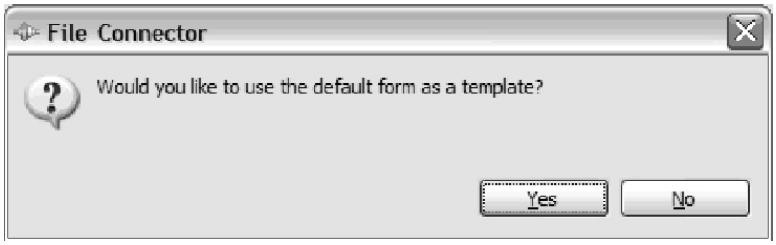


Figura 62. Menu de Contexto - Opção Editor de Formulários

Observe que quando você escolhe um editor diferente do padrão para um arquivo, o CE se lembrará de sua escolha, de modo que na próxima vez em que você der um clique duplo no arquivo, ele abrirá o arquivo com o editor usado por último.

Para abrir o componente com o editor padrão, basta escolher o editor apropriado neste menu (geralmente o primeiro editor da lista).

Se o componente que você abrir não tiver nenhum formulário customizado, será solicitado o preenchimento do formulário com o formulário padrão:



Escolher **Sim** criará uma definição no formulário inicial com base no formulário padrão do componente. Nesse caso, o conector do FileSystem é usado no exemplo, o que resulta na seguinte tela:

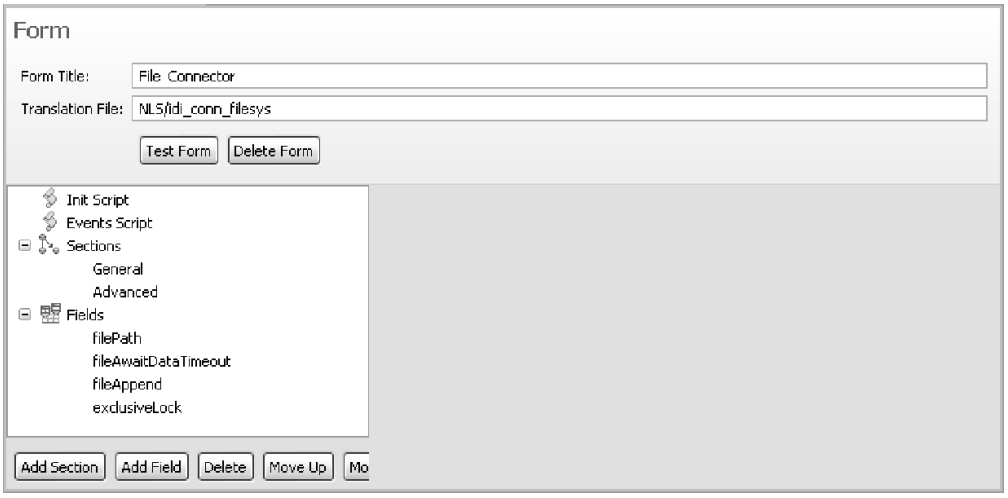


Figura 63. Tela Editor de Formulários Padrão do Conector FileSystem

Os diversos elementos vistos neste formulário têm a seguinte função:

Título do Formulário
Este é o título principal do formulário do componente (deixe em branco se não houver título). Isso é o que o usuário vê na parte superior do formulário customizado.

Arquivo de Tradução
Esse é o arquivo usado para traduzir as etiquetas e as dicas de ferramentas do formulário. Todas as etiquetas e dicas de ferramentas tentarão ser traduzidos se você definir um arquivo de tradução. Se você criar uma etiqueta com “file_name” como o texto, a tradução tentará recuperar uma sequência do arquivo de tradução usando “file_name” como a chave. Os próprios arquivos de tradução são arquivos de propriedade simples com “key=value” em cada linha.

Para localizar um formulário, você deve criar arquivos com o identificador de código do idioma. Assim, para uma tradução de francês, você deveria criar um arquivo chamado *base-name*`_fr.properties`.

Formulário de Teste
Esse botão mostrará uma janela de diálogo com a definição de formulário atual.

Excluir Formulário
Esse botão excluirá o formulário do componente. Você deve fechar e escolher **Salvar** para que isso tenha efeito permanente.

Script de Inicialização
O script de inicialização é executado quando o formulário é carregado. Isso é onde você coloca o código para inicializar o estado do formulário e quaisquer variáveis do script global do formulário.

Script de Eventos
Quando um valor de campo é alterado, o Formulário executará um manipulador de eventos definido neste script. Cada campo tem um nome interno que o componente usa. Por exemplo, o FileConnector usa "filePath" como seu nome interno para o parâmetro **Caminho de arquivo**. Você pode ver todos os nomes de parâmetros do componente quando optar por preencher seu formulário com o formulário padrão da seção Campos. Para reagir às alterações no formulário, você grava os manipuladores de eventos no editor de script de eventos usando o nome interno com o sufixo "_changed" como o nome da função de script. Abaixo, você verá um exemplo do conector LDAP que desativa dois campos com base no método de autenticação:

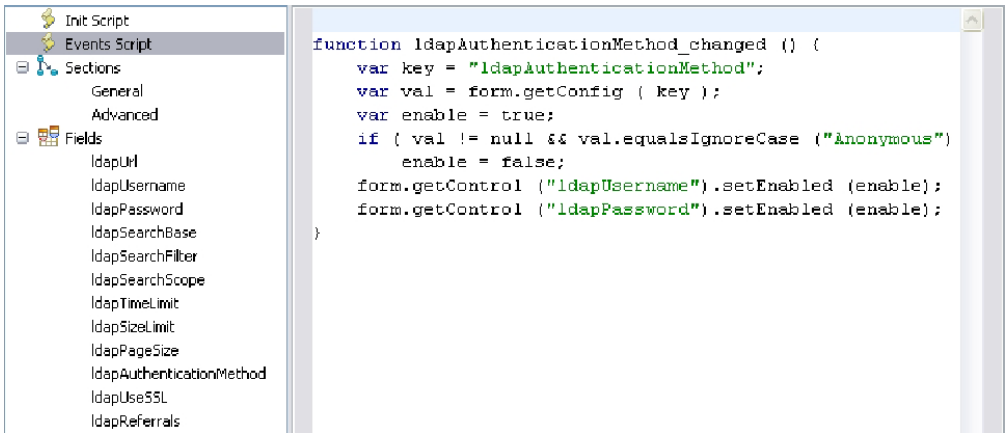


Figura 64. Editor de Formulários, Script de Eventos no Conector LDAP

Seções e Campos
Seções e campos são o que compõem o formulário. Você gerencia as seções e os campos incluindo, removendo e reorganizando a ordem dos mesmos na barra de ferramentas abaixo da árvore.

- **Incluir Seção** – inclui uma nova seção vazia no formulário
- **Incluir Campo** – inclui um novo campo no formulário. Os nomes de campos devem ser exclusivos.
- **Excluir** – remove uma seção ou um campo do formulário.
- **Mover Para Cima/Para Baixo** – reorganiza a ordem das seções e dos formulários. Quando você possui seções definidas, a ordem dos campos

é definida pela seção e não pela lista de campos. Quando não houver seções definidas, a ordem desta árvore determinará a ordem dos campos no formulário.

Seções

Esta parte é opcional. Se você não especificar seções, então o formulário terá todos os campos exibidos em seu formulário, todos de uma vez.

As Seções são usadas para organizar campos no formulário. As Seções são como pastas que o usuário pode expandir e reduzir para exibir/ocultar seu conteúdo. Quando você define uma seção, especifica se a seção é inicialmente expandida e quais os campos que devem ser exibidos. No exemplo do conector FileSystem, há duas seções.

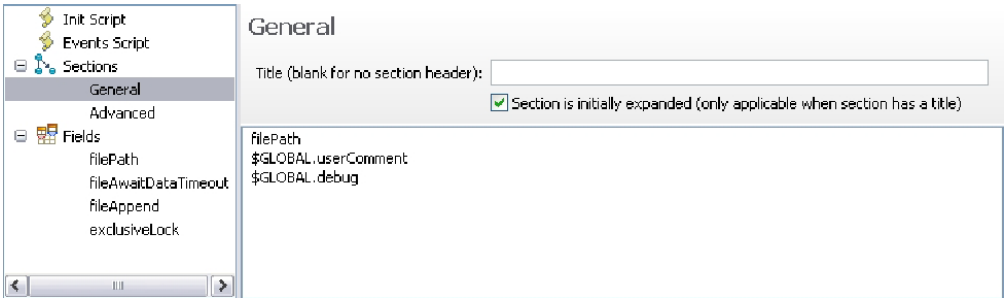
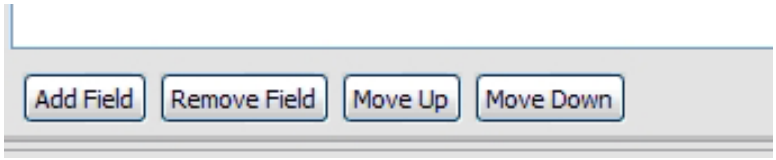


Figura 65. Editor de Formulários - Seção Geral

A primeira seção está na seção *Geral*. Essa seção não possui título, o que significa que não deve haver um cabeçalho da seção em que o usuário possa clicar para expandir ou reduzir seu conteúdo. Isso a torna uma seção estática, uma vez que a seção não pode ser reduzida nem expandida. É possível incluir, remover e reordenar os campos na lista de campos usando a barra de ferramentas na parte inferior do painel:



A segunda seção é a seção *Avançada*:

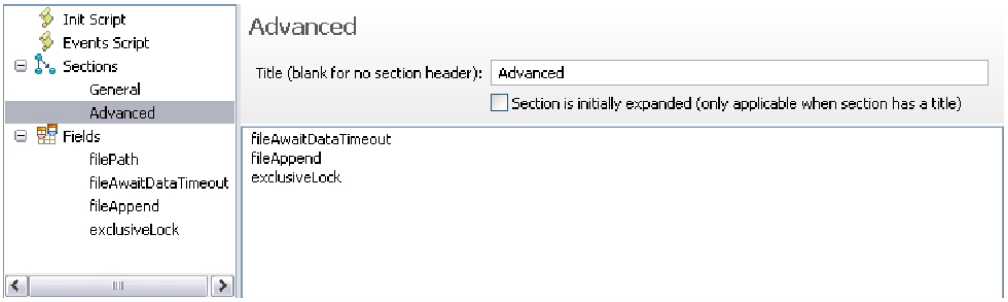


Figura 66. Editor de Formulários - Seção Avançada

Esta seção possui um título, mas ele não é expandido inicialmente. Isso fará com que o formulário seja exibido com esse título de seção reduzido e os campos dentro dele ficarão ocultos até que o usuário expanda a seção.

Campos

Campos são os parâmetros do componente. Se você reutilizar um componente como o conector FileSystem, deverá fornecer os parâmetros necessários no formulário ou definir o parâmetro da seção "Script de Inicialização" de modo que o componente possa funcionar corretamente. Cada campo tem sua definição mostrada quando você o seleciona.

Figura 67. Editor de Formulários - Definições de Campos

Neste painel, você verá a definição do parâmetro do conector "filePath". Na ordem de aparição:

Tabela 9. Editor de Formulários - definição de parâmetros

Campo	Descrição
Rótulo	O rótulo desse formulário será mostrado
Dica de Ferramenta	A dica de ferramenta exibida quando o usuário passa o mouse sobre o campo de entrada
Tipo de Campo	O tipo de campo de entrada: • Sequência para entrada de texto em uma única linha • Suspenso (editável) para um campo de entrada suspenso editável • Suspenso (não-editável) para uma lista suspensa com um conjunto fixo de seleções • Booleano para uma caixa de opção • Área de Texto para um campo de entrada de texto em várias linhas • Texto Estático para exibição de texto simples (ou seja, sem entrada) • Senha para um campo de entrada em uma única linha protegido por senha • Editor de Scripts para edição de scripts • Componente Customizado para um controle SWT/JFace definido pelo usuário
Seleção de Modos	Esse campo opcional pode especificar modos de componentes em que o campo é excluído ou incluído. Especifique modos separados por uma vírgula com um sinal de menos para excluir. "Repetidor" – mostrado apenas no modo Repetidor "-Repetidor" – Não mostrado no modo Repetidor "Repetidor,Consulta" – mostrado apenas no modo Repetidor e Consulta

As três guias na parte inferior permitem especificar botões, valores suspensos para listas Suspensas e o nome da classe Java para o componente customizado.

Assistentes

Há vários Assistentes (procedimentos com etapas assistidas graficamente) no IBM Security Directory Integrator Config Editor. São eles:

1. "Assistente para Importar Configuração"
2. "Assistente de Novo Componente" na página 142
3. "Características do Formulário de Configuração do Conector" na página 148

Assistente para Importar Configuração

É possível importar arquivos de configuração de versões anteriores usando o assistente Importar Configuração.

Nesse assistente, você escolhe o projeto de destino e quais componentes deseja importar.

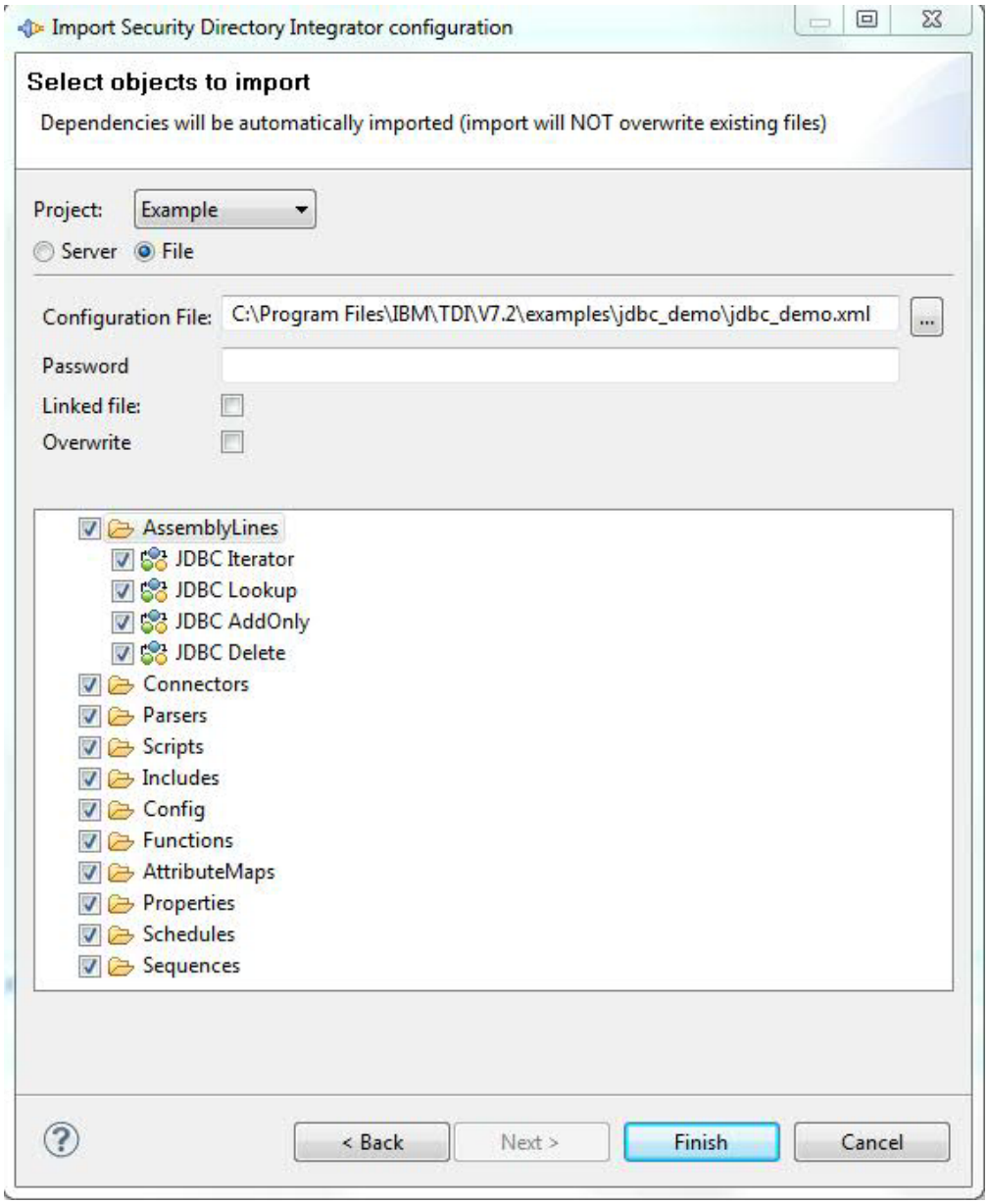


Figura 68. Assistente para Importar Configuração

Por padrão, todos os componentes são selecionados para importação. Entretanto, você pode marcar apenas aqueles de seu interesse. Se você marcar um único AssemblyLine e esse AssemblyLine usar conectores no arquivo de configuração, os conectores também serão importados automaticamente.

O campo de entrada **Projeto** é o projeto de destino para o qual a configuração é importada. Selecione a opção em branco para criar um novo projeto.

O campo de entrada **Arquivo de Configuração** é o arquivo de configuração que você importará. Se a configuração for protegida por senha, insira a senha no campo de entrada **Senha**.

Quando o campo **Arquivo Vinculado** é marcado, quaisquer mudanças feitas no projeto importado são gravadas de volta no arquivo do qual o projeto foi importado. É possível alterar essa configuração nas propriedades do projeto limpando ou alterando o nome do arquivo para o campo **Arquivo Vinculado**, conforme ilustrado abaixo:

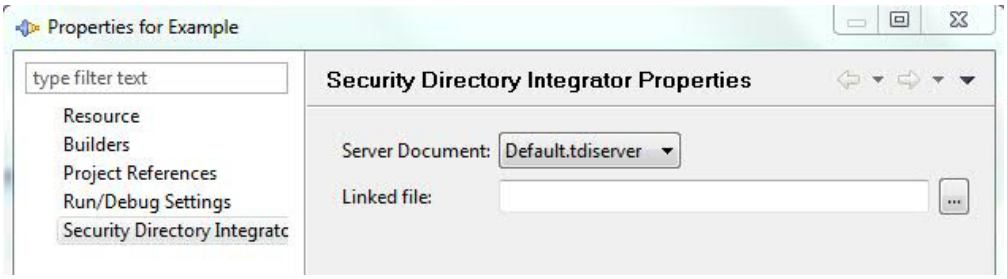


Figura 69. Campo Arquivo Vinculado

Também é possível importar configurações dos servidores. Alterne para a visualização do servidor marcando o botão de opções **Servidor**.

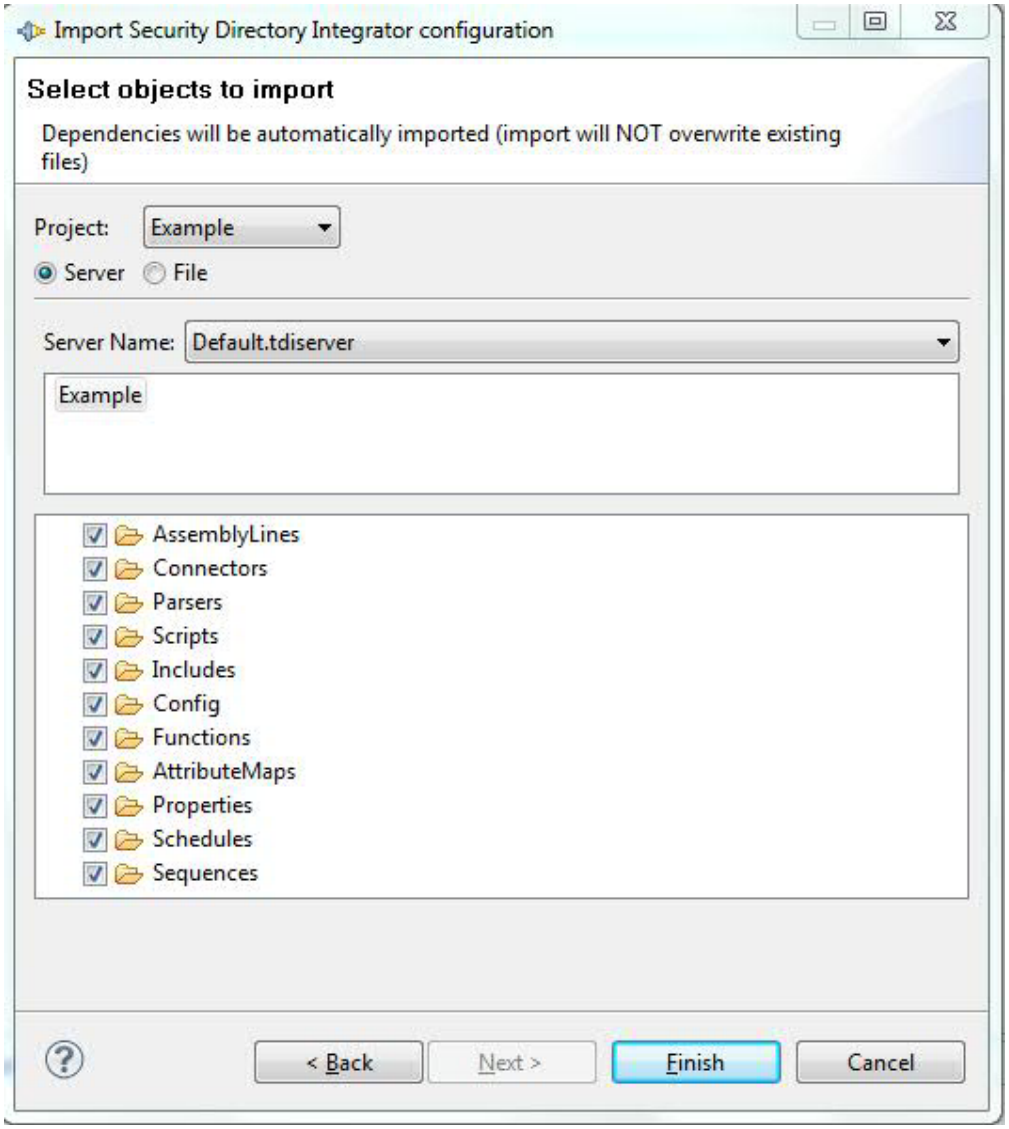


Figura 70. Assistente "Importar do Servidor"

Você inicia esse tipo de importação selecionando o servidor da lista de servidores no campo **Nome do Servidor**. Após você ter selecionado um servidor, a lista de configurações nesses servidor será mostrada na lista abaixo do nome do servidor. Como essa é uma operação de rede, talvez você veja alguns dos controles desativados enquanto a lista está sendo atualizada. Na lista de configurações, você seleciona a configuração que quer importar. A seleção de uma configuração faz o download dessa configuração a partir do servidor e preenche a árvore abaixo com seu conteúdo para que você possa escolher quais componentes incluir na importação.

Assistente de Novo Componente

Esse assistente é chamado quando você usa **Incluir Componente** em um AssemblyLine ou quando usa **Arquivo** > **Novo...** na barra de menus principal.

Ao criar um novo componente na pasta `Recursos`, você terá um layout de assistente um pouco diferente. Para um novo conector, por exemplo, o assistente é semelhante a este:

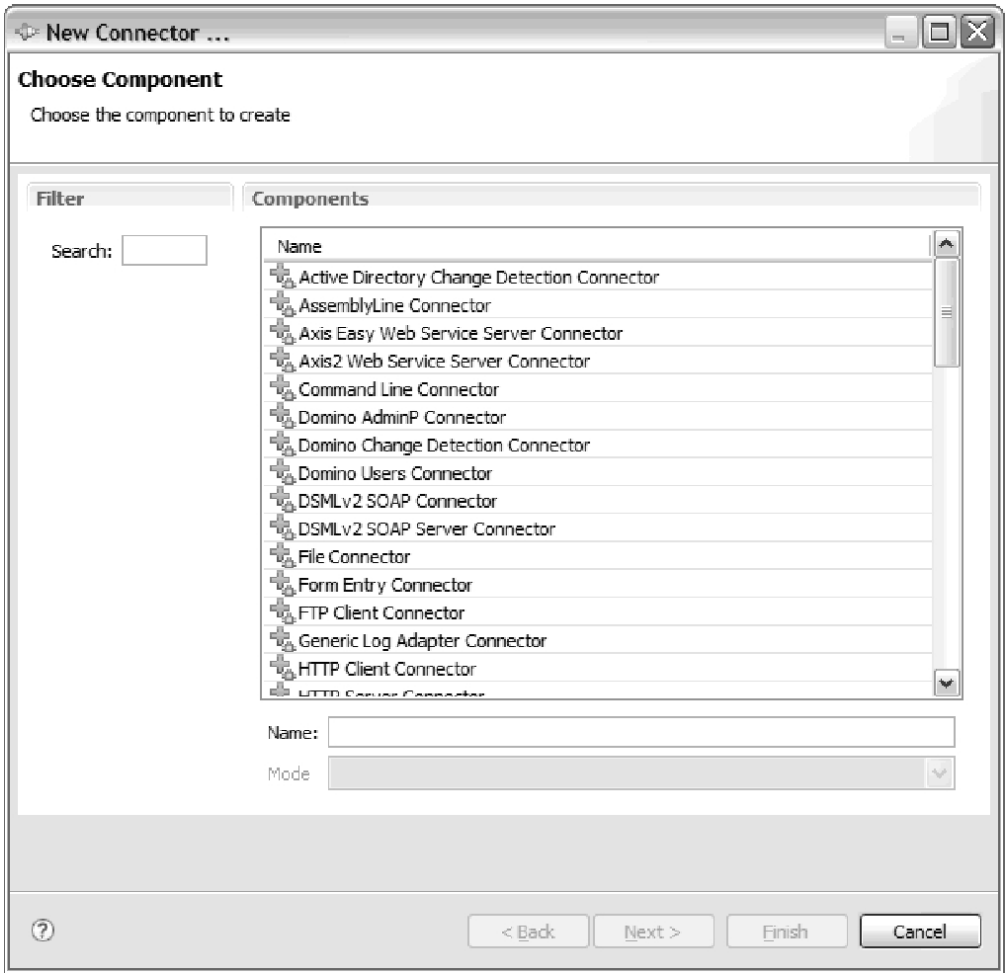


Figura 71. Assistente de Novo Conector na Pasta de Recursos

O nome do componente será usado como o nome do arquivo sugerido na pasta; é aconselhável alterar esse nome para algo significativo.

Ao criar um novo componente em um AssemblyLine, você terá várias outras opções nesse assistente.

A primeira página do assistente é a página de seleção de tipo de componente. Nessa página, você escolhe o tipo de componente que deseja incluir ou criar. O lado esquerdo contém uma lista de filtros que selecionarão componentes relevantes com base no rótulo na lista.

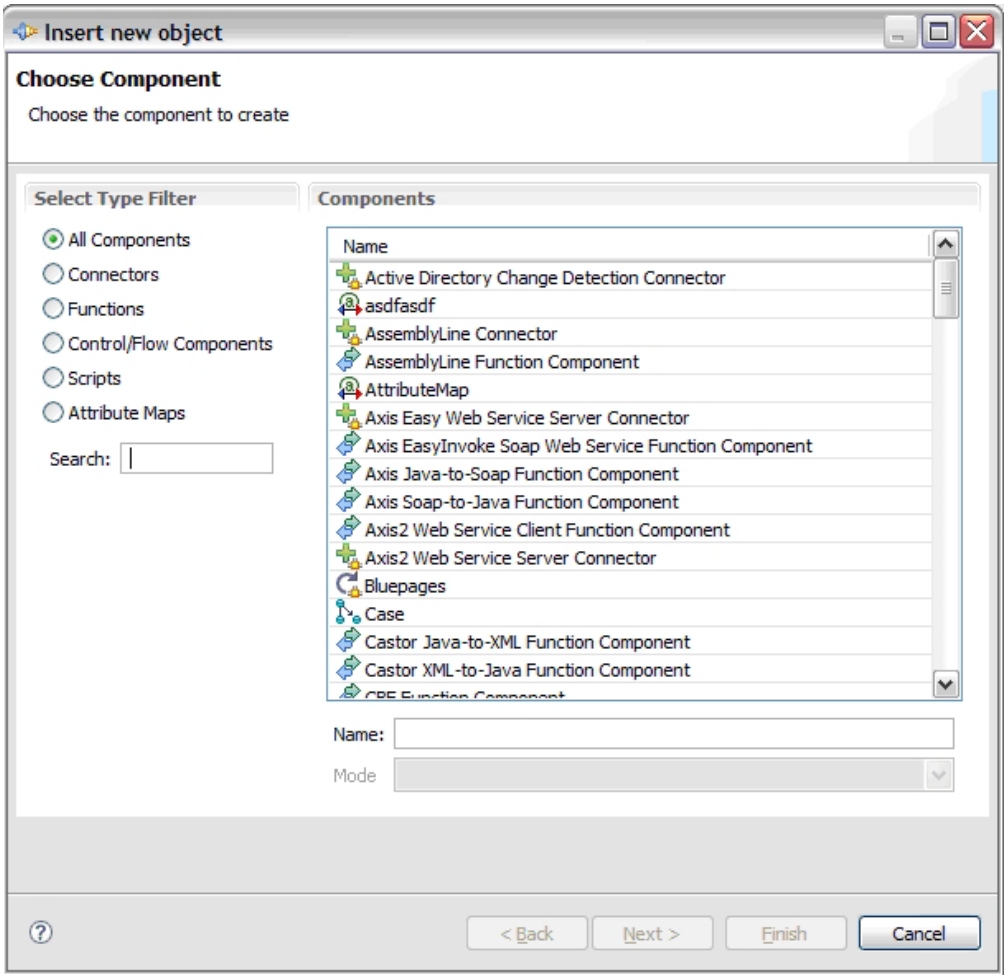


Figura 72. Assistente de Novo Componente

É possível filtrar o conteúdo digitando no campo Procurar. Conforme você digitar, a lista será correspondida com relação ao que foi digitado no campo de procura (sem distinção entre maiúsculas e minúsculas contém correspondência).

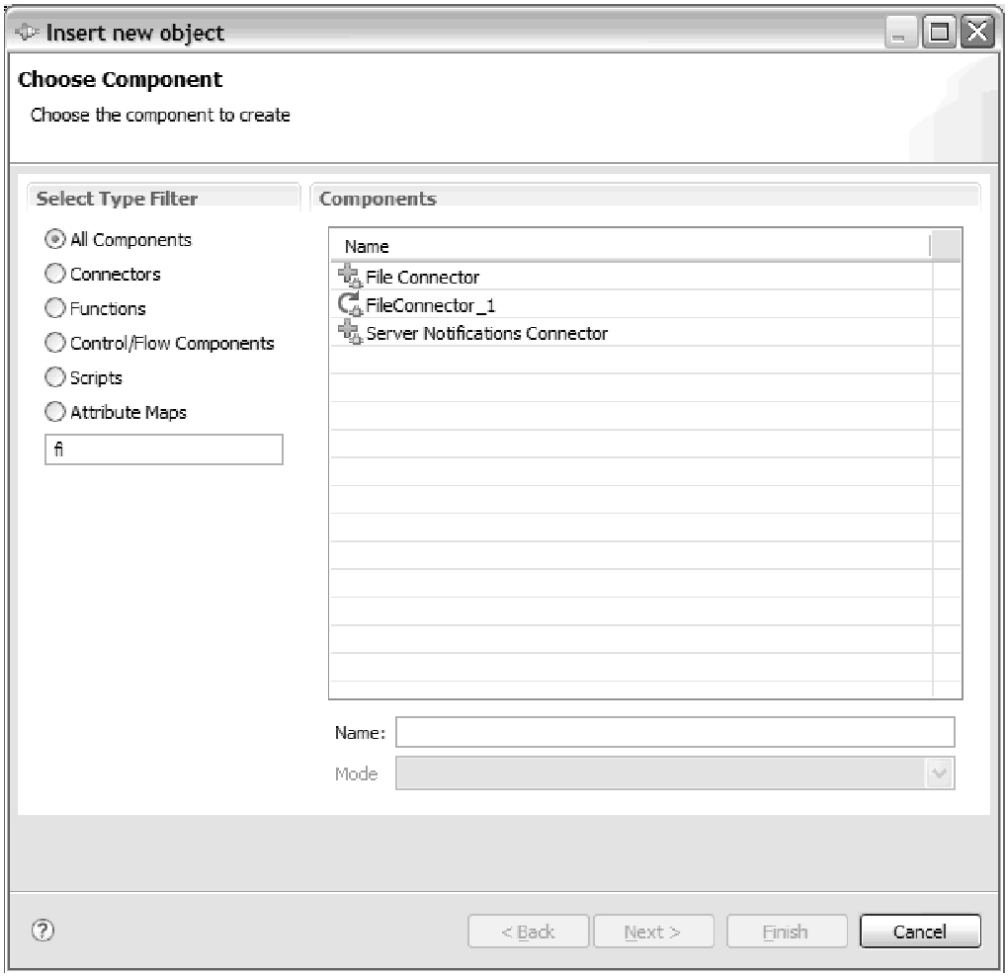


Figura 73. Assistente de Novo Componente, com Filtragem

Neste ponto, você poderá optar por concluir o assistente e o componente será inserido em seu AssemblyLine.

Ao escolher um conector, uma série de formulários serão apresentados a você para definir apropriadamente o conector. Para todos os outros tipos, você pode apenas concluir o assistente e configurar o componente no AssemblyLine.

Após selecionar o tipo, o painel de configuração do Conector será apresentado.

Figura 74. Painel de Configuração do Conector

Consulte também “Características do Formulário de Configuração do Conector” na página 148 para obter algumas informações específicas sobre como completar esse tipo de formulário.

A próxima etapa mostrará a configuração do Analisador se o conector puder usar um.

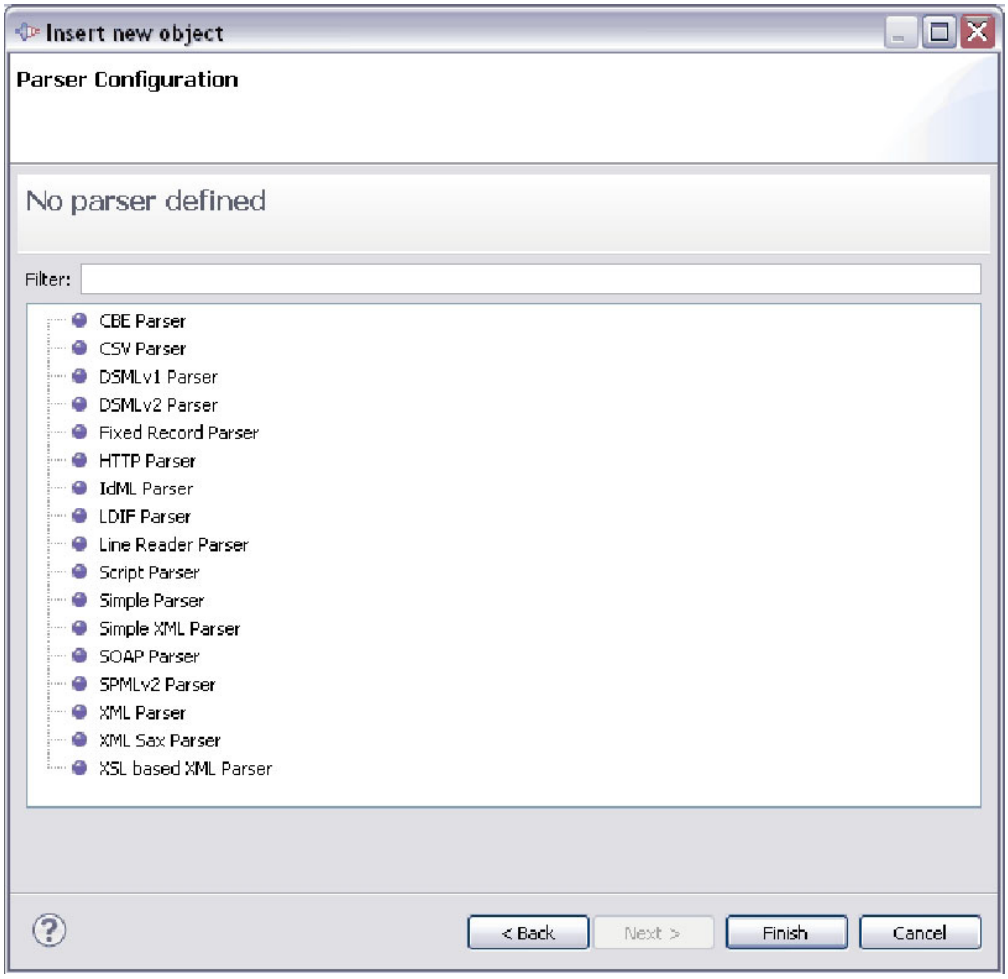


Figura 75. Painel de Configuração do Analisador

Use o botão **Selecionar Analisador** para escolher o analisador para o componente:

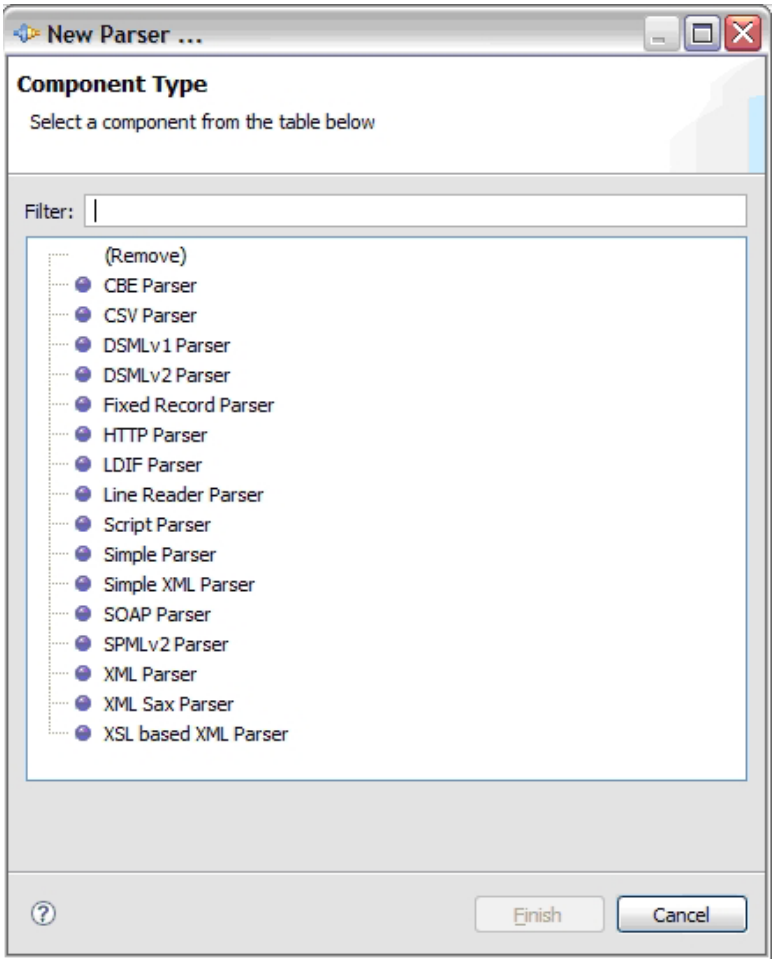


Figura 76. Diálogo Seleção de Analisador

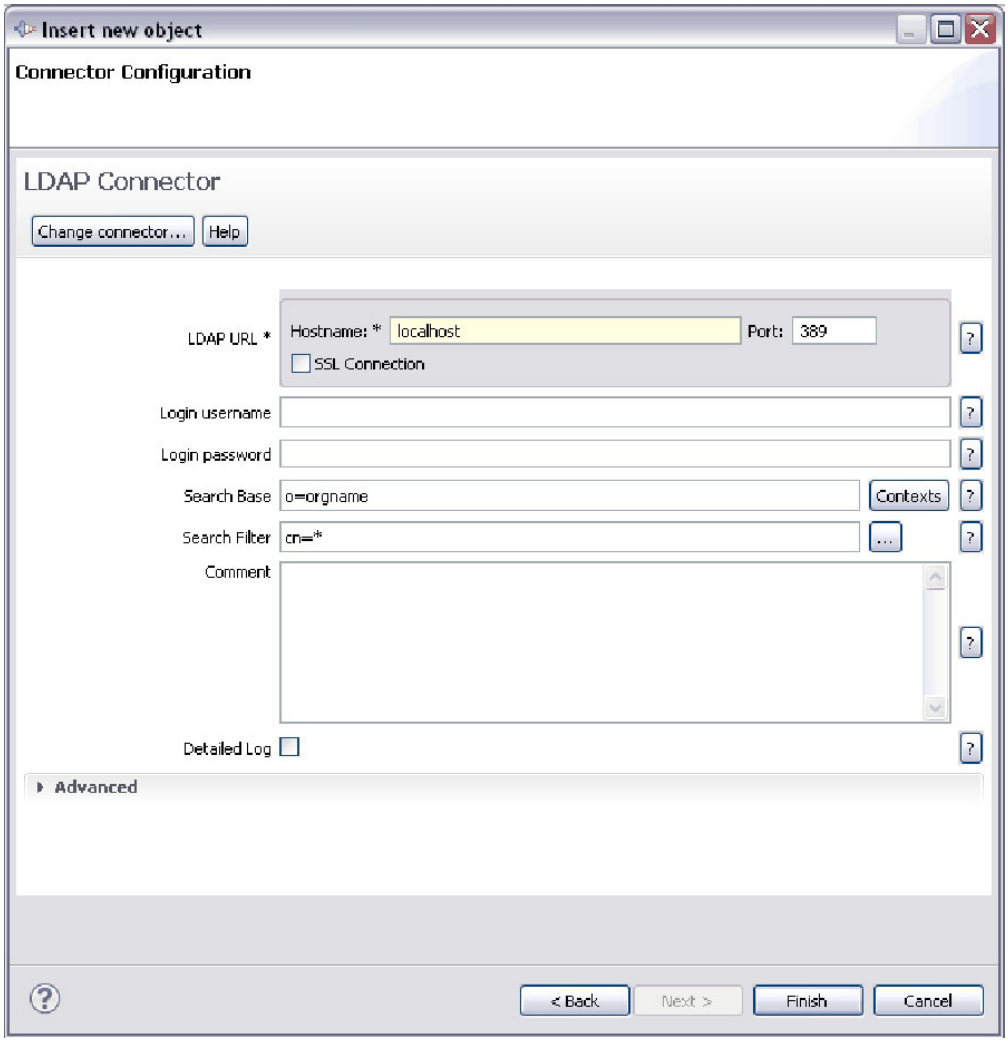
Você pode remover o analisador atual do componente, selecionando a opção "(Remover)".

Características do Formulário de Configuração do Conector

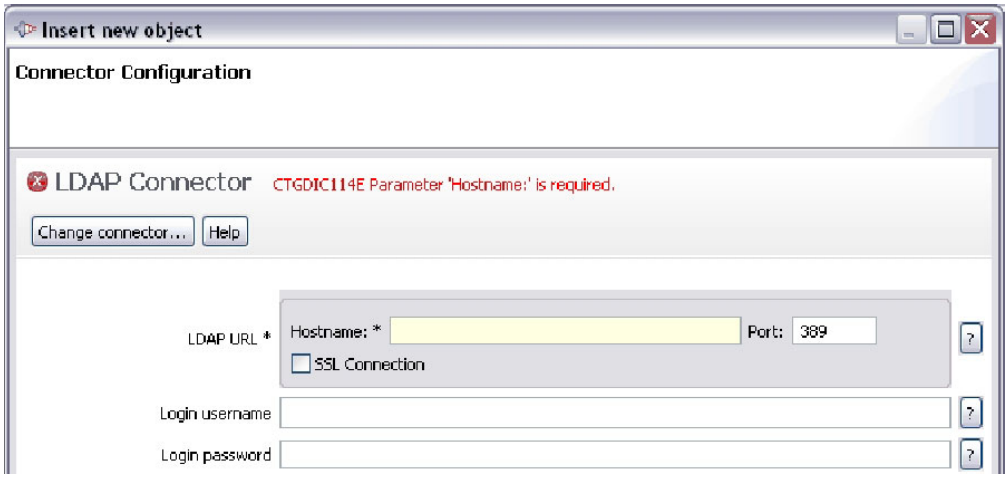
Os campos em um formulário podem ser agrupados em seções intituladas opcionalmente e em campos que tenham uma propriedade que especifica se o campo é obrigatório.

Na janela Configuração do Conector, os campos obrigatórios são indicados por um * após o nome do campo.

Os valores herdados terão um rótulo azul.



Na amostra acima, a configuração do conector LDAP mostra duas seções com um campo obrigatório. Quando um campo obrigatório não tiver um valor, o formulário marcará isso no título.



Quando você passa o mouse sobre o ícone ou o texto em vermelho, verá todas as mensagens de erro e a quais campos os erros se aplicam. Isso será útil quando houver mais de um erro no formulário.

Executando e Depurando AssemblyLines

O Editor de Configuração está equipado com vários mecanismos que ajudam você a desenvolver AssemblyLines, incluindo recursos para testar e depurar a lógica deles.

Relatórios do AssemblyLine

Os Relatórios do AssemblyLine podem ser executados a partir do menu de contexto no navegador.

Clique com o botão direito do mouse em um AssemblyLine e escolha o submenu **Criar Relatório do AssemblyLine**. Esse menu contém todos os modelos de relatório que estão no diretório *TDI_Install_dir*/XSLT/ConfigReport.

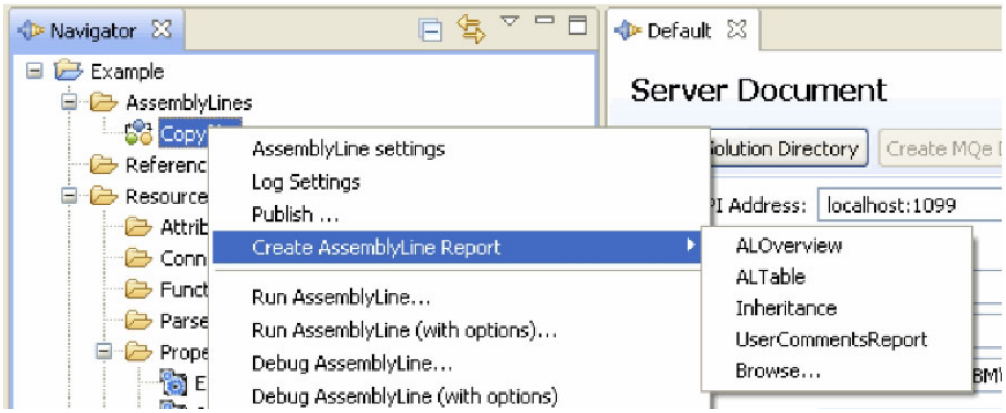


Figura 77. Comando "Criar Relatório do AssemblyLine"

Escolha a opção **Pesquisar...** para pesquisar pelo sistema de arquivos local de um modelo de relatório.

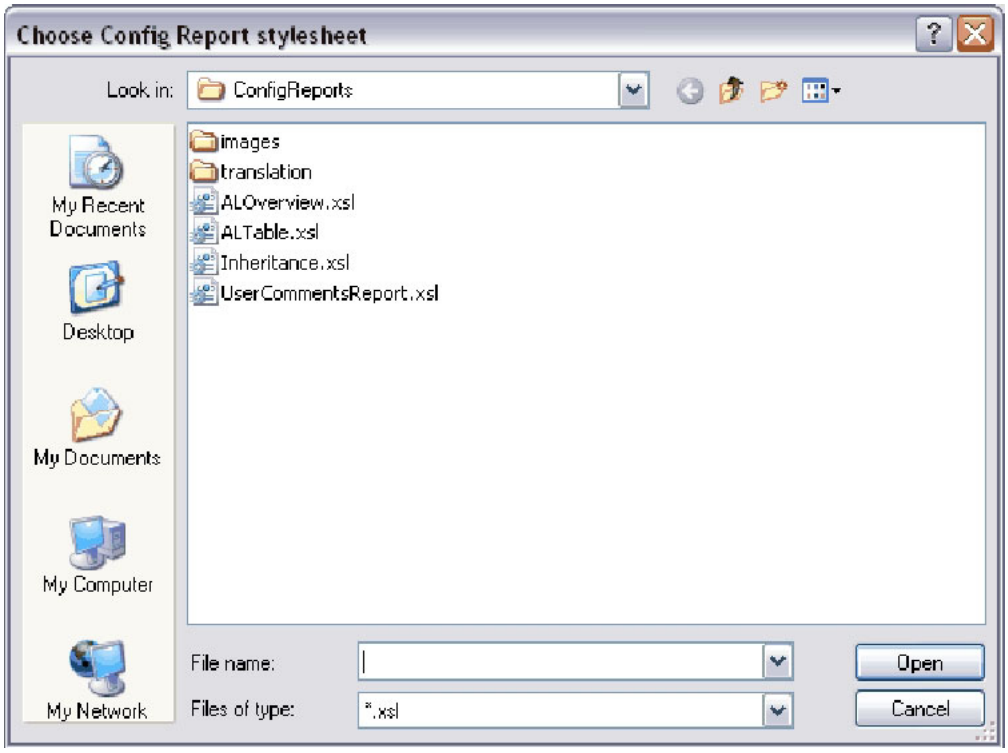


Figura 78. Diálogo Escolher Folha de Estilo do Relatório de Configuração

Quando você seleciona uma e clica em **Abrir**, o relatório é gerado e colocado no diretório `Relatórios` de seu projeto, conforme visto na próxima figura. O editor associado à extensão de arquivo *.html* é aberto para visualizar o relatório, geralmente seu navegador de internet padrão.

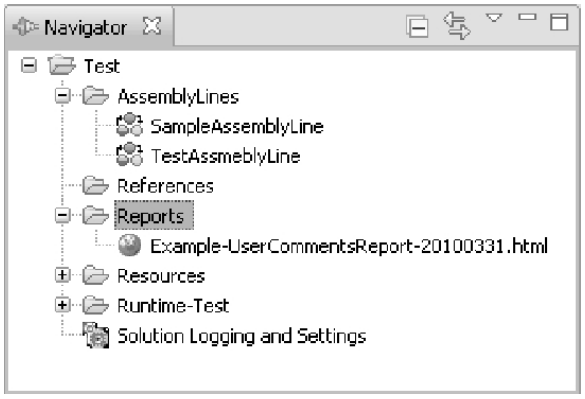
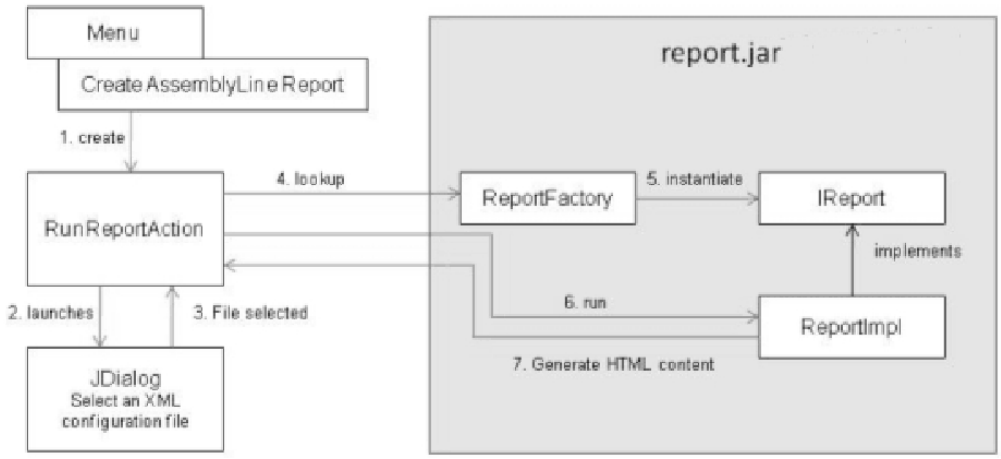


Figura 79. Pasta Relatórios da Hierarquia Projeto

O relatório do AssemblyLine gera nomes de arquivos de relatório com base no AssemblyLine com a data atual inserida.

Visão Geral de Relatórios do AssemblyLine Baseados em XML

É possível gerar um relatório para o elemento de configuração selecionado com base em uma folha de estilo do relatório ou para o arquivo XML de configuração de relatório especificado. O diagrama a seguir representa a arquitetura de relatórios do AssemblyLine baseados em XML.



Formato do Arquivo XML de Configuração do Relatório

O arquivo XML de configuração do relatório do AssemblyLine possui o seguinte formato:

```
<tdiReport>
<reportClass)com.ibm.di.report.aloverview.AssemblyLineOverview</reportClass>
    <reportConfig>
        <report specific configuration>
    </reportConfig>
</ tdiReport>
```

O arquivo XML de configuração possui os seguintes elementos:

- **reportClass** - especifica o nome de classe Java para o relatório.
- **ReportFactory** - instancia uma instância do relatório.
- **reportConfig** - contém parâmetros específicos do relatório.
-

Executando o AssemblyLine

À medida que você desenvolve o AssemblyLine, é possível testá-lo executando até a conclusão ou por etapas por meio dos componentes, um a um.

Há dois botões para executar o AssemblyLine. O primeiro botão (reproduzir ícone, ) inicia o AssemblyLine e mostra a saída em uma visualização de console. O segundo botão (**Depurador**) executa o AssemblyLine com o depurador.

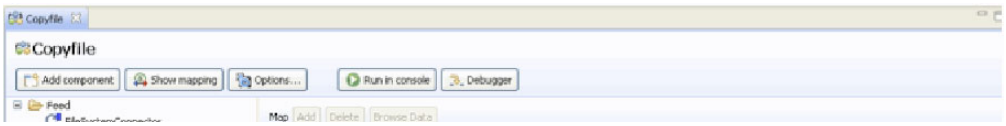
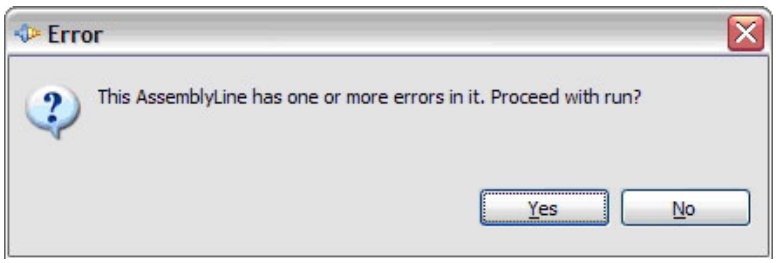


Figura 80. Três Opções para Iniciar um AssemblyLine

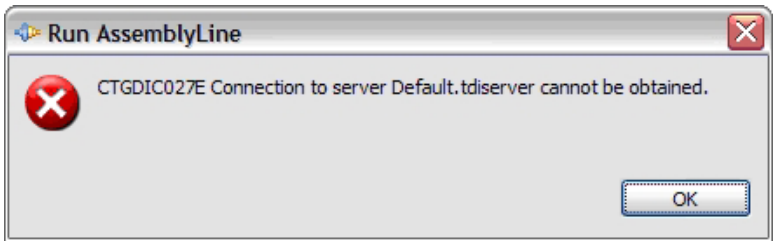
O processo de iniciar um AssemblyLine tem três etapas.

Se o AssemblyLine contiver erros (como mapas de saída ausentes, etc.), será solicitada a confirmação da execução do AssemblyLine:

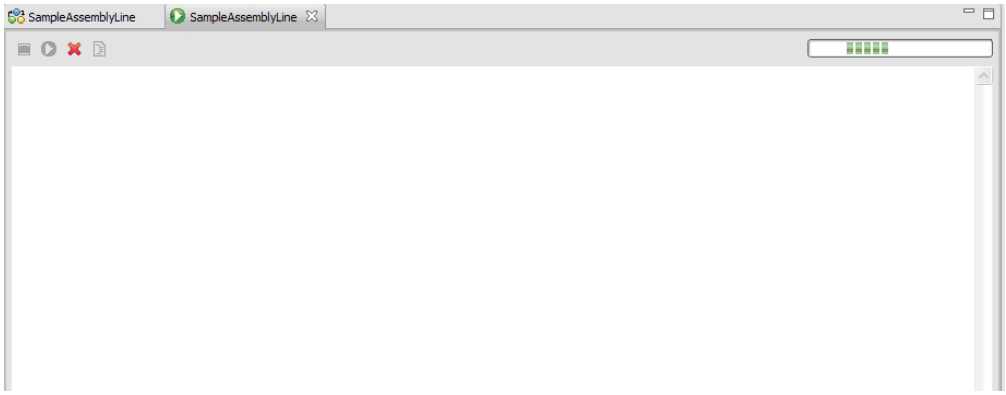


Se você obtiver esse diálogo, deverá verificar a visualização Problemas para ver quais erros estão possivelmente interrompendo o AssemblyLine. Geralmente, durante o desenvolvimento, você fica ciente disso e ainda deseja executar o AssemblyLine; nesse caso, pressione Enter ou clique no botão **Sim** para executar o AssemblyLine.

A próxima verificação é se o servidor IBM Security Directory Integrator está disponível. Se o servidor não puder ser atingido, você verá esta mensagem:

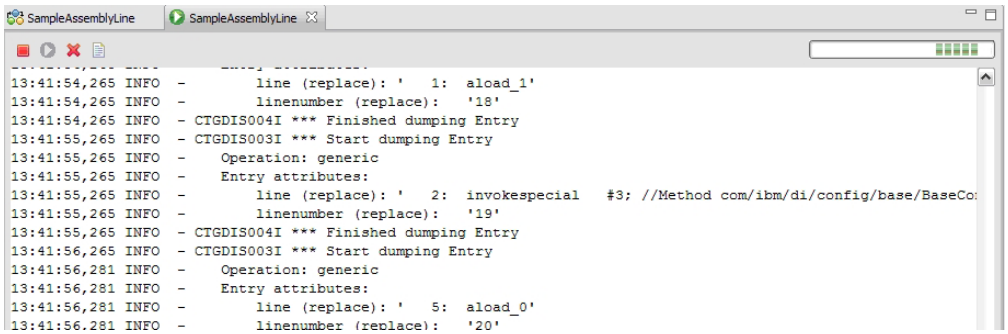


Ao executar um AssemblyLine a partir do CE, a primeira etapa será quando o CE transfere a configuração de tempo de execução para o servidor e aguarda até que o AssemblyLine seja iniciado. Nesta etapa, você verá uma barra de progresso na parte superior da janela. O botão da barra de ferramentas para parar o AssemblyLine também estará esmaecido, uma vez que ele não foi iniciado ainda.



A segunda etapa é quando o AssemblyLine está em execução. A barra de progresso estará girando e você começará a ver as mensagens na janela Log. Agora, você pode parar o AssemblyLine pressionando o botão Parar (à esquerda) na barra de ferramentas.

Nota: O botão Parar funcionará apenas se o servidor tiver controle na execução do encadeamento. Se o encadeamento estiver executando alguma coisa fora do IBM Security Directory Integrator, clicar no botão Parar pode não ter muito efeito.



Se você tiver várias janelas abertas do AssemblyLine, poderá informar quais dos AssemblyLines correspondentes estão em execução, pois seus nomes terão um prefixo igual a '*' (asterisco).

Quando o AssemblyLine tiver parado (normalmente ou pressionando o botão Parar), a barra de progresso desaparecerá e o item da barra de ferramentas para executar novamente o AssemblyLine estará ativado. O botão Parar está agora desativado, uma vez que o AssemblyLine não está em execução.

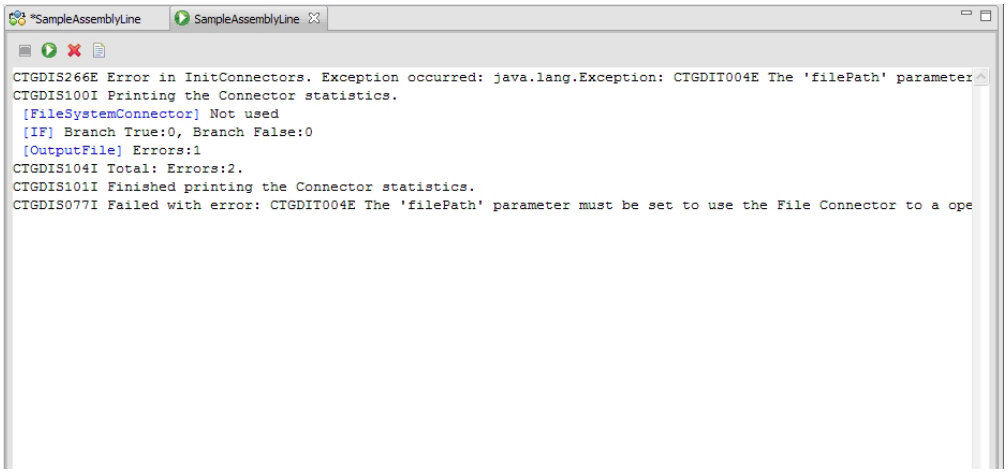


Figura 81. Janela Log do Console

A qualquer momento, você pode limpar a janela Log. A janela Log mostra apenas as últimas centenas de linhas do log do AssemblyLine, mas cada mensagem de log é gravada em um arquivo de log temporário, de modo que você pode abrir o arquivo de log em uma janela do editor separada usando o botão Exibir log da barra de ferramentas (à direita).

Nota: É possível alterar o tamanho do buffer do log subjacente à janela Log nas 300 linhas padrão para outro valor, acessando **Janela** > **Preferências** > **Preferências do Security Directory Integrator** > **Número máximo de linhas para a janela Executar AssemblyLine**.

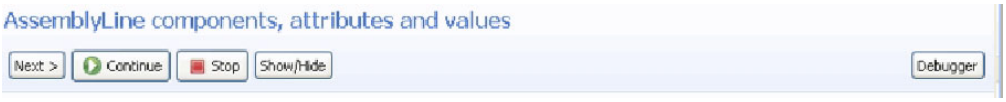
Quando o AssemblyLine tiver terminado, você poderá executar o AssemblyLine novamente com o botão Executar novamente.

Observe o texto em azul na janela Log. Quando você vir esse texto, poderá manter pressionada a tecla CTRL enquanto clica na palavra que o levará a essa parte do AssemblyLine.

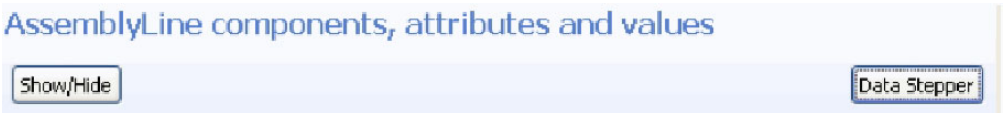
O Stepper e o Depurador

O stepper e o depurador são ferramentas integradas ao Configuration Editor e podem ajudar você a desenvolver AssemblyLines interativamente.

É possível executar o stepper em dois modos. Um é o depurador avançado normal (ativado pelo botão **Depurador**; consulte "O Depurador" na página 157) onde você tem acesso a todas as partes do AssemblyLine; o outro é o stepper (ativado pelo botão **Data Stepper**; consulte "O Data Stepper") que fornece uma visualização mais simples dos componentes e do fluxo. É possível alternar entre os dois clicando no botão na visualização de coluna:



Na visualização de stepper, é possível alternar para a visualização de depurador clicando no botão **Depurador**. Do contrário, para alternar do depurador para o stepper, você clica no botão **Data Stepper**.



O Data Stepper

O data stepper fornece uma visualização de coluna de todos os conectores no AssemblyLine. Percorrendo o AssemblyLine, você verá os dados lidos ou gravados para cada componente. Todos os dados são mostrados nessas tabelas após um conector ter concluído sua operação.

Figura 82. Janela Principal do Data Stepper

Na parte direita, o data stepper mostra componentes com um mapa de atributo verticalmente. Cada componente pode ser removido da visualização clicando no

botão fechar ou cancelando sua seleção no diálogo **Mostrar/Ocultar**. O botão à esquerda do botão fechar () em cada componente é um atalho para "Executar Aqui".

Os botões **Avançar** e **Executar** são utilizados para percorrer um componente por vez ou para executar o AssemblyLine até a conclusão. O botão **Parar** é utilizado para pausar um AssemblyLine em execução ou finalizar um AssemblyLine pausado. A parte esquerda do data stepper mostra o esboço para o AssemblyLine e a entrada de trabalho abaixo. Na visualização do esboço, você pode optar por reativar o AssemblyLine a partir do menu de contexto. Isso irá iniciar o depurador em uma nova sessão e executá-lo até o componente selecionado. Essa é uma maneira rápida de chegar ao depurador a partir do data stepper.

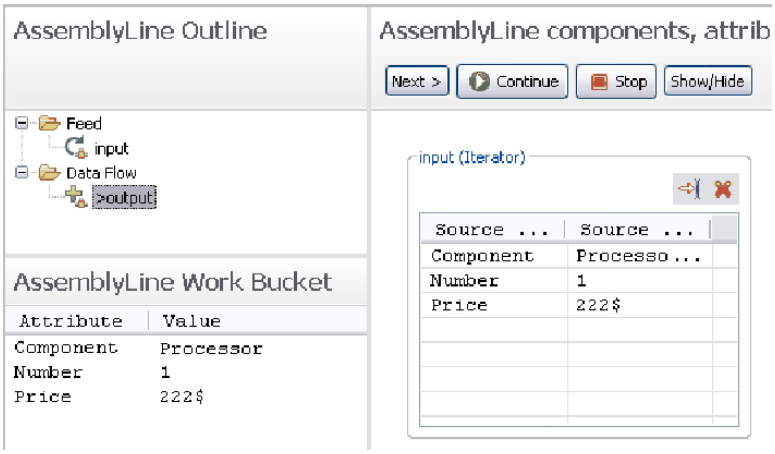


Figura 83. Botão **Mostrar/Ocultar** *no Data Stepper*

O diálogo **Mostrar/Ocultar** Componentes permite que você escolha quais componentes mostrar na visualização.

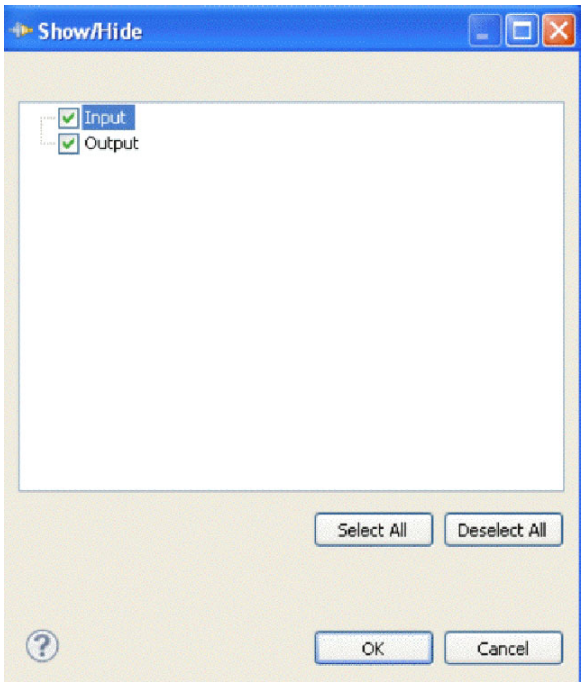


Figura 84. Diálogo Mostrar/Ocultar Componentes

O Depurador

Quando escolher a visualização **Depurador**, você será apresentado a um layout muito semelhante à visualização do stepper, mas na visualização do depurador, você vê muito mais dos componentes do AssemblyLine, além da janela de procura na qual é possível ter expressões customizadas. A árvore de componentes do AssemblyLine tem uma caixa de seleção para cada item que você pode marcar ou desmarcar para configurar ou remover um ponto de interrupção. Além disso, existem mais botões de comando que permitem acessar componentes e ganchos.

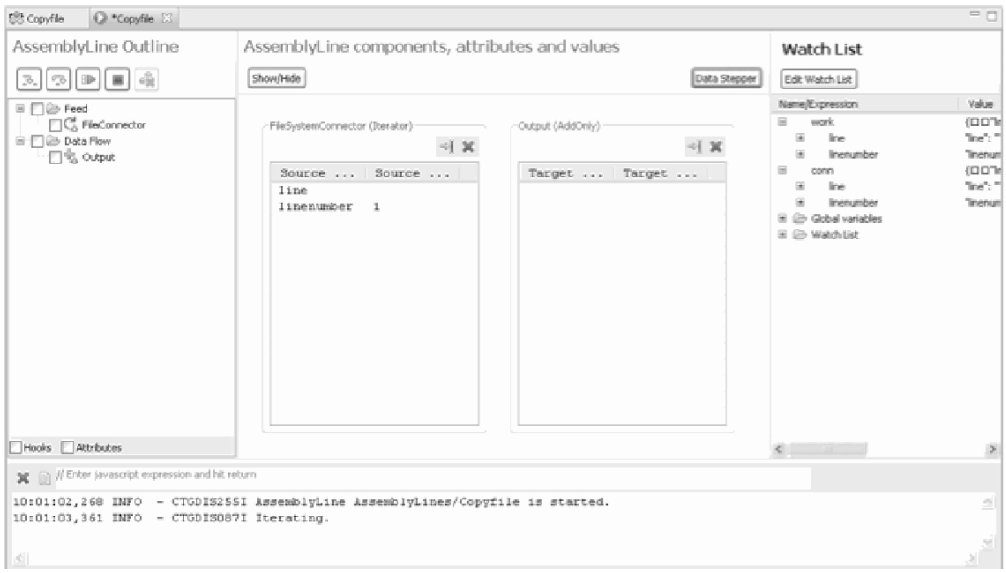


Figura 85. Janela do Depurador

(Na janela Depurador, pode ser útil maximizar o editor com Ctrl-M para obter uma visão geral melhor).

A árvore à esquerda mostra os componentes e ganhos do AssemblyLine. Você pode alternar a caixa de opção de cada item para definir um ponto de interrupção nesse ponto. Dê um clique duplo no item para visualizar o script ou para inserir o script em uma quebra condicional.

A imagem abaixo mostra a guia da quebra condicional após o clique duplo no gancho **Antes de GetNext**. Além disso, observe que você pode ocultar todos os ganchos que estão inativos. Mostrar todos os ganchos permite definir um ponto de interrupção, independentemente de o gancho estar ou não ativo. A caixa de opção **Atributos** permite ocultar mapas de atributos da visualização em árvore.

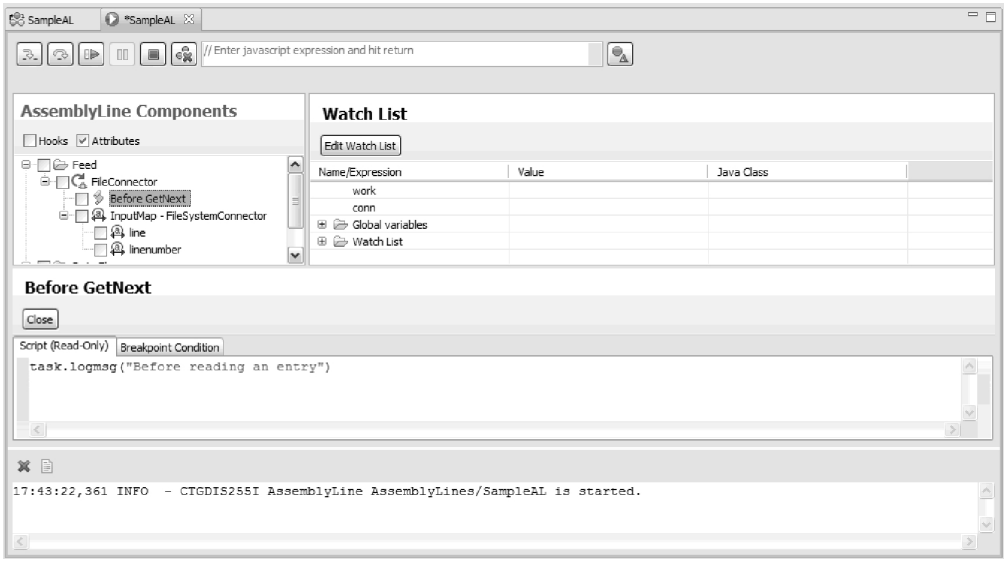


Figura 86. Depurador Antes de GetNext

O painel Mapa de atributos mostra os componentes e suas designações e também o valor designado para o atributo de trabalho. O valor é uma captura instantânea da última interrupção do AssemblyLine e o **Valor Anterior** é o valor da captura instantânea antes dela. Quando você avança pelo AssemblyLine, os valores se refletirão nos mapas e scripts que afetam a entrada de trabalho.

Quando ocorrer um erro, o stepper mostrará a mensagem de exceção e o rastreio de pilha em um diálogo separado. O arquivo de log (na tela) não inclui esse rastreio de pilha. Você pode optar por não mostrar esse diálogo selecionando a caixa de seleção ou na página de preferências do Configuration Editor.

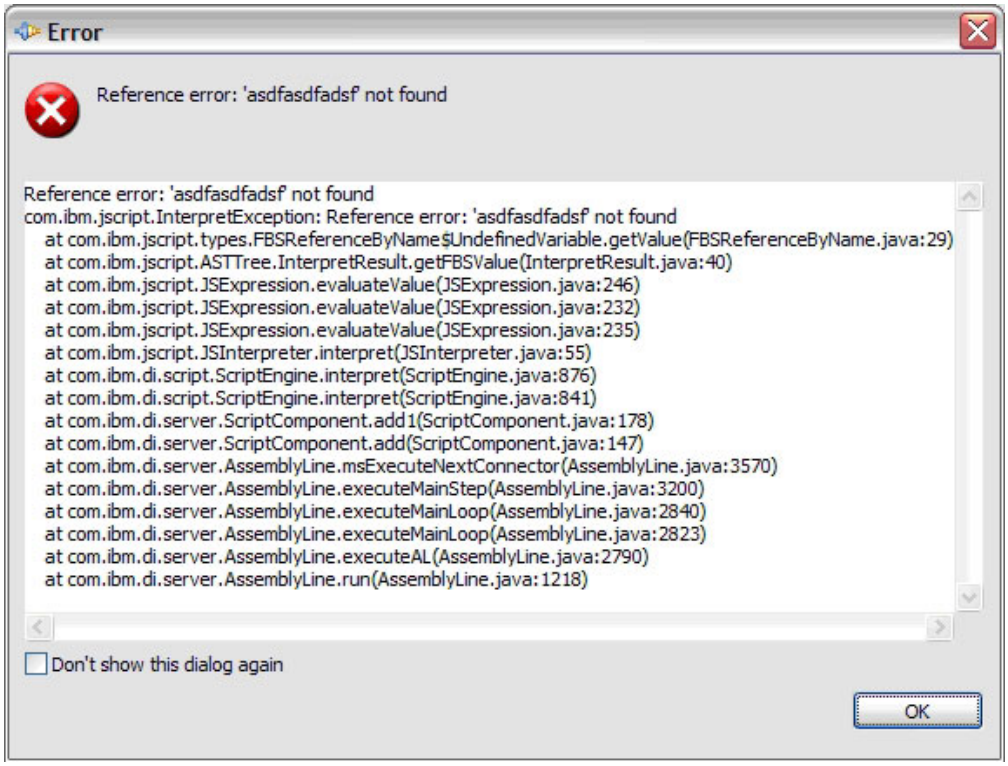


Figura 87. Diálogo de Erro: Rastreio de Pilha

Avançando pelos Scripts

Quando atinge um ponto de interrupção que contém JavaScript, você pode entrar no script e executá-lo linha por linha. A guia Script mostrará o script prestes a ser executado; você pode seguir o fluxo usando o comando **Entrar na Etapa** (à esquerda dos dois botões Stepper).

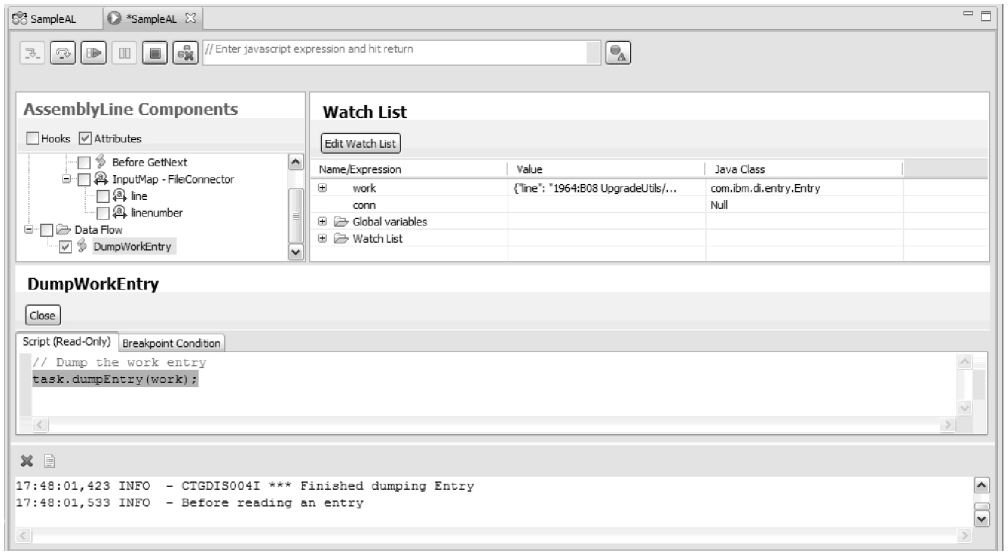


Figura 88. Janela Depurador: Avançando um Script Linha por Linha

À medida que você avança pelo script, cada linha será destacada antes de ser executada. Você pode também usar a função **Avaliar** para exibir variáveis de mecanismo do script enquanto avança por um script.

Quando uma função de script está prestes a ser executada, é possível usar o botão **StepInto** para entrar na função JavaScript. Se a função for definida fora do contexto atual (por exemplo, gancho, script de mapa de atributo), a janela será substituída.

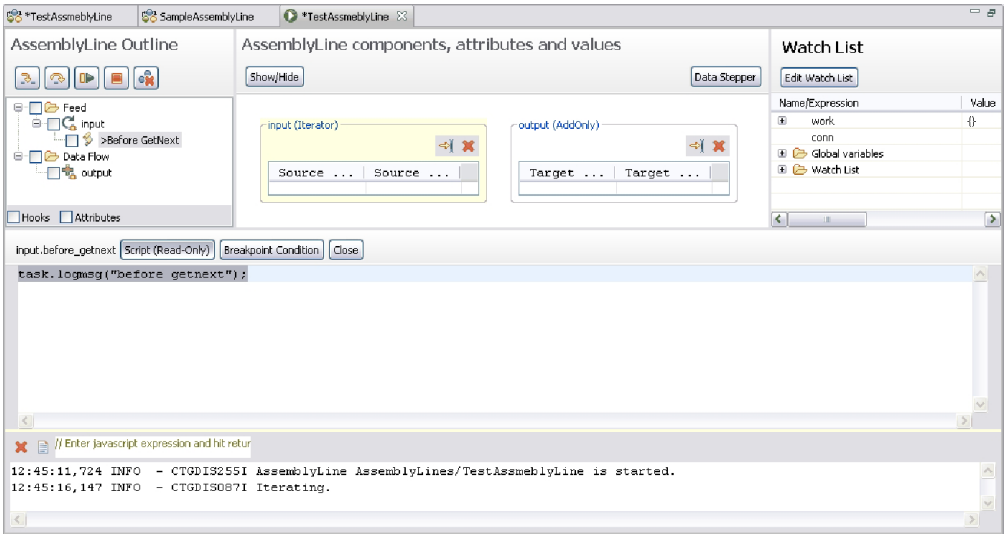


Figura 89. Função StepInto

Neste AssemblyLine, definimos uma função chamada `myfunc1()` no gancho "antes de inicializar". Nós o chamaremos a partir do componente Script1 para mostrar como ele aparece para você:

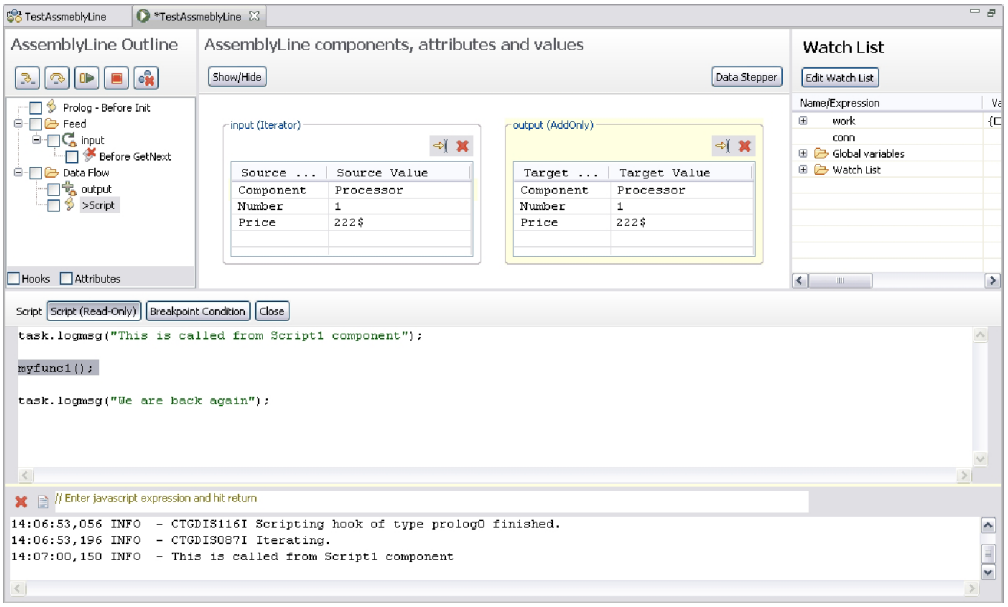


Figura 90. Função Stepped-into

Neste ponto, podemos pressionar **StepOver** para continuar com a próxima instrução ou pressionar **StepInto** para seguir a chamada da função JavaScript:

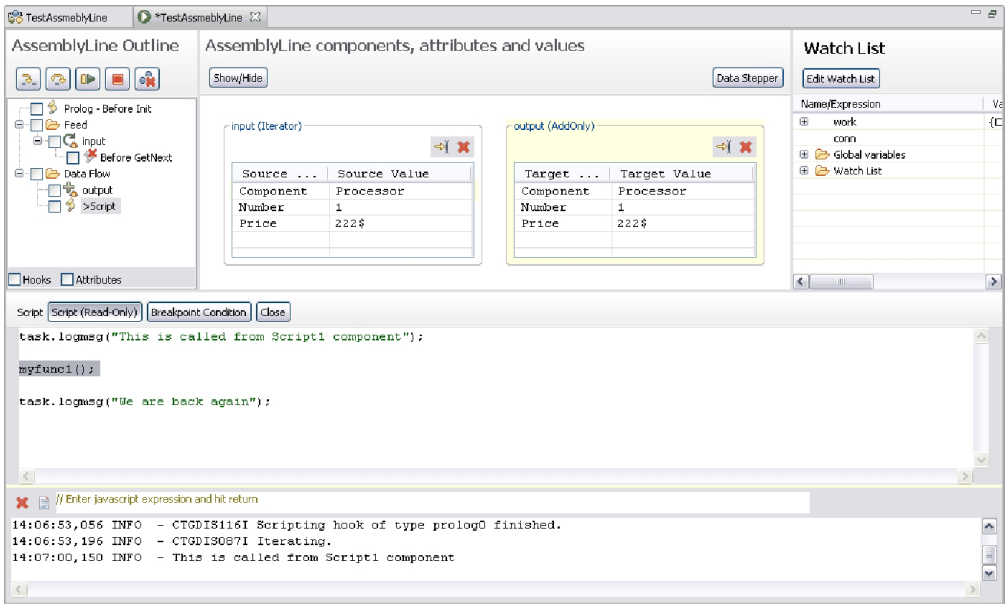
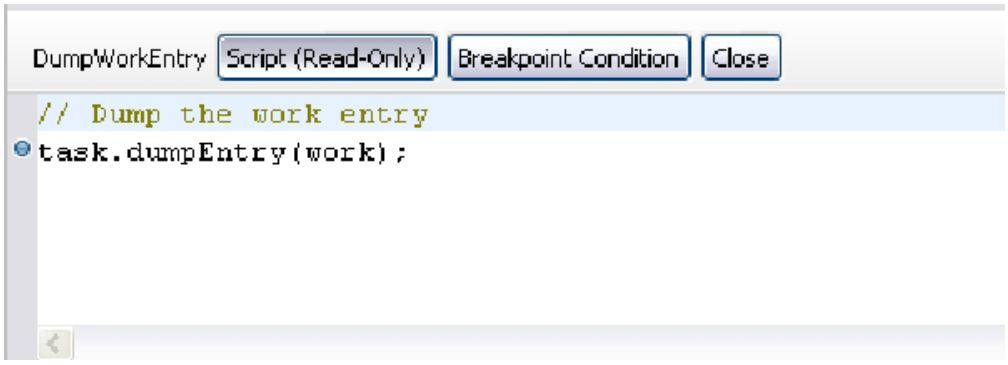


Figura 91. Seguir a Chamada de Função

Step into faz o editor de script mudar para o script a partir do gancho *antes de inicializar*. Observe a etiqueta alterada que mostra onde o script está definido.

Também é possível definir pontos de interrupção dentro dos scripts. Abra o editor para um script ou mapa de atributos e dê um clique duplo na margem esquerda. Um marcador azul aparecerá para denotar que existe um ponto de interrupção definido para esse local. Dê um clique duplo novamente para removê-lo. Você pode também clicar com o botão direito do mouse na margem esquerda ou no campo de texto para alterar os pontos de interrupção.



Depuração do Servidor

Depurar um servidor IBM Security Directory Integrator significa que cada AssemblyLine iniciado em um servidor IBM Security Directory Integrator estabelecerá automaticamente uma sessão de depuração com o Configuration Editor como se ele fosse iniciado no modo de etapa.

A depuração do servidor é ativada selecionando-se **Depurar Servidor** no menu suspenso em um servidor da visualização Servidores. Quando você escolhe essa opção, o CE se conecta ao servidor e configura uma propriedade Java apontando

para o CE para depuração.

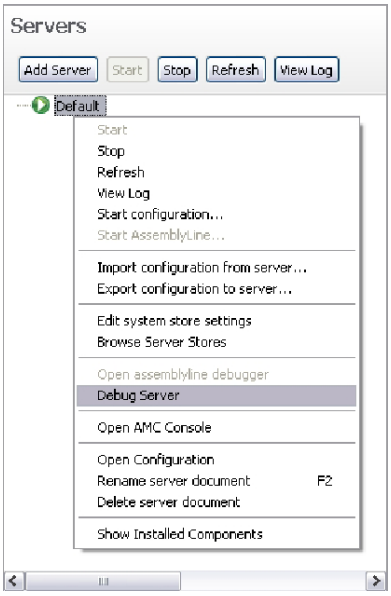
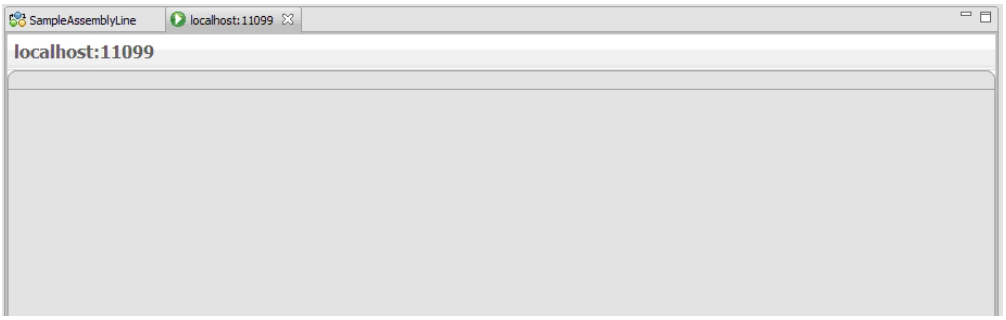


Figura 92. Opção Depurar Servidor

Selecionar **Depurar Servidor** inicia uma nova janela em que os AssemblyLines aparecem à medida que são iniciados no servidor. Isso é diferente de iniciar ativamente um AssemblyLine no CE. Quando você inicia uma sessão de depuração no CE, ele terá sua própria janela e não aparecerá nesta janela de depuração do servidor.



Cada AssemblyLine iniciado por outro CE ou componente no servidor de destino tem seu próprio painel de stepper dentro desta janela. Consulte a seção “O Stepper e o Depurador” na página 155 para obter uma descrição do painel de stepper.

Opções de Execução

Você pode especificar opções adicionais ao executar um AssemblyLine. Essas opções são salvas de modo que sempre que você executar o AssemblyLine, ele usará essas opções.

Você pode selecionar o modo de execução, a operação do AssemblyLine e fornecer uma entrada de trabalho inicial (IWE) para o AssemblyLine.

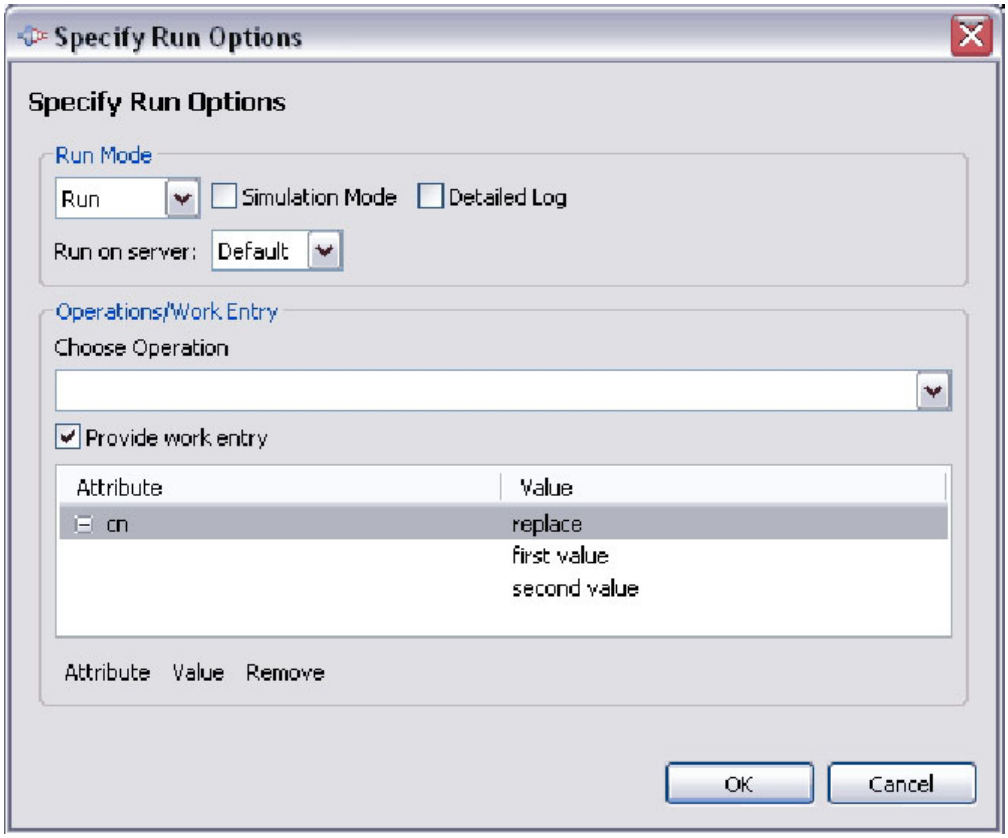


Figura 93. Janela Executar com Opções

Use os botões **Atributo** e **Valor** para incluir atributos e valores à entrada de trabalho inicial.

Escolhendo o Servidor

Quando você executa um AssemblyLine, ele será executado no servidor de desenvolvimento local. Esse servidor é iniciado pelo CE e seu diretório de soluções está localizado no projeto *Servidores TDI*. Quando você cria um novo servidor do IBM Security Directory Integrator, é possível alterar o servidor preferencial de um determinado projeto por meio do diálogo **Propriedades** do projeto:

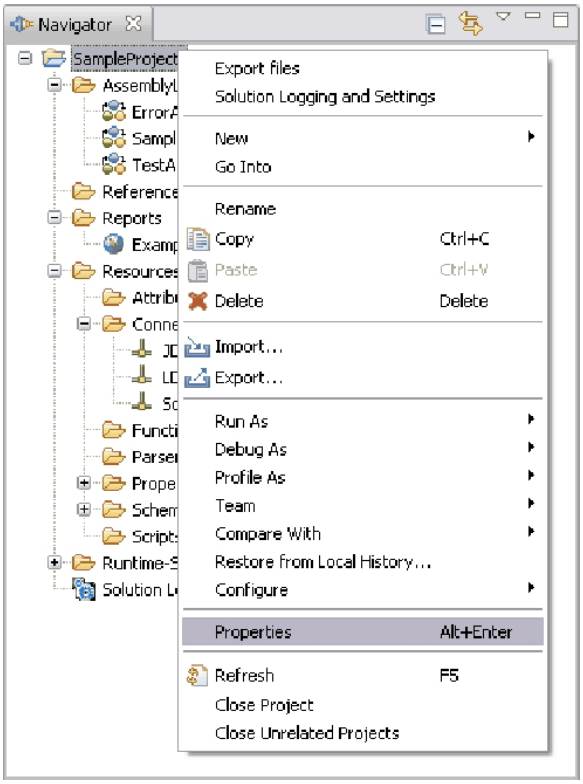


Figura 94. Opção de Menu Propriedades do Projeto

Selecionar esse item abrirá o diálogo de propriedades do projeto. Selecione **Propriedades do Directory Integrator** para alterar o servidor padrão do projeto.

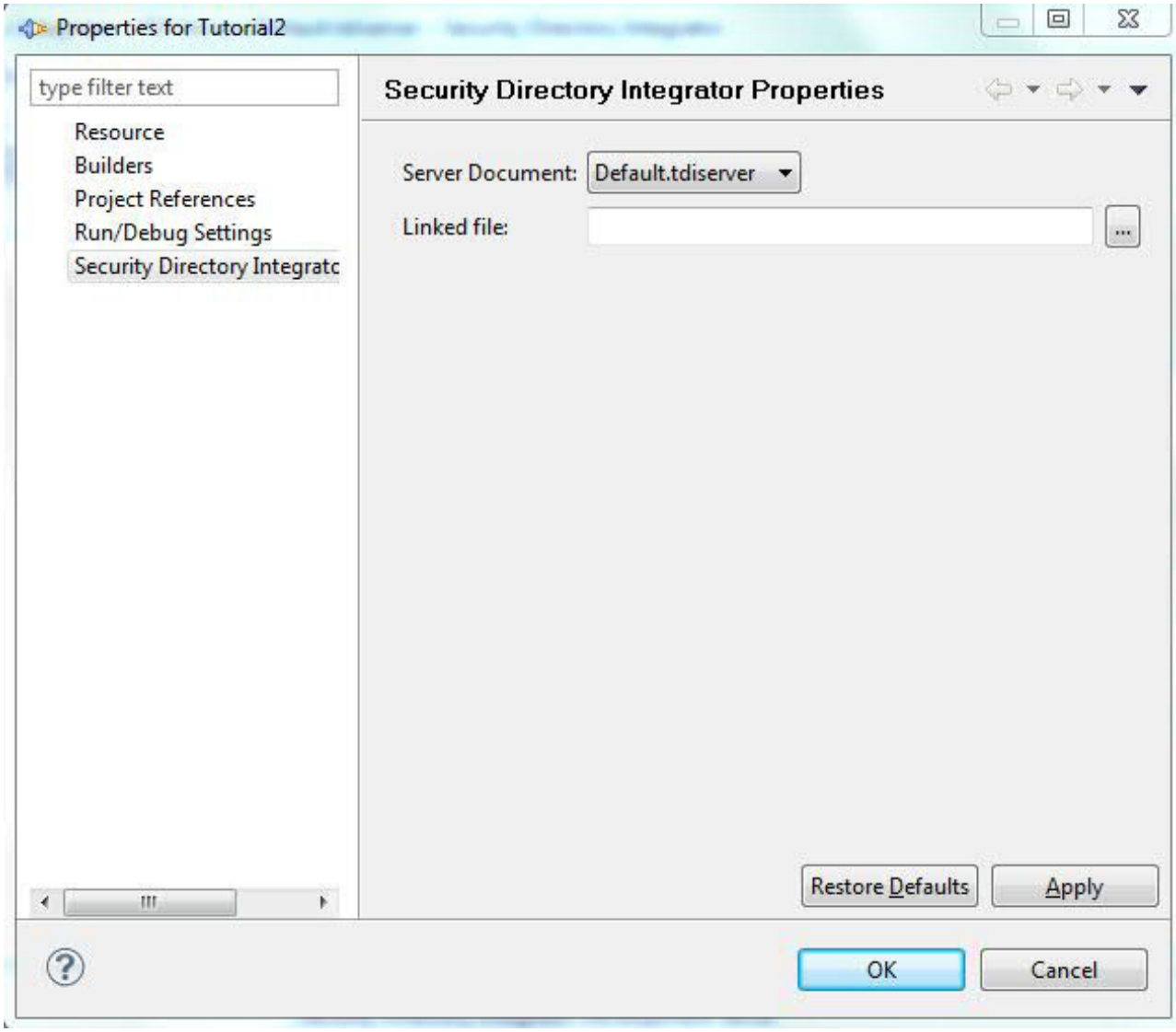


Figura 95. Propriedades do Projeto

Equipe de Suporte

O Editor de Configuração do IBM Security Directory Integrator inclui os plug-ins do Eclipse que ativam o compartilhamento de projetos entre usuários usando um repositório de controle de código de origem.

IBM Security Directory Integrator inclui bibliotecas CVS que suportam pserver, pserverssh2, e conexões do tipo ext e extssh. Para obter informações mais detalhadas sobre os plug-ins CVS do Eclipse, consulte o site CVS do Eclipse em http://www.eclipse.org/eclipse/platform-cvs.

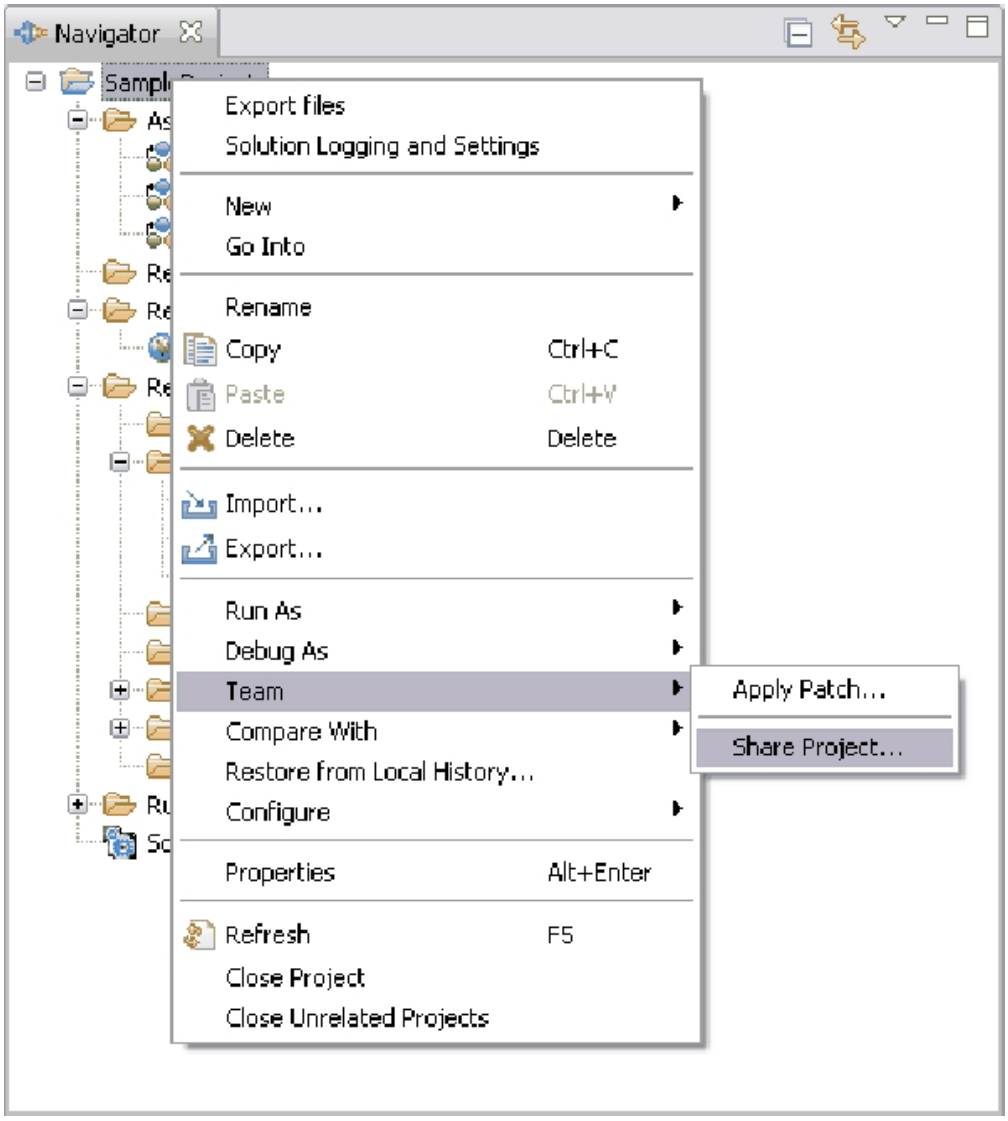
Para usar os recursos de compartilhamento da equipe, você precisa acessar um repositório CVS. Os repositórios CVS podem ser hospedados pela maioria dos sistemas operacionais e pacotes binários estão geralmente disponíveis na rede. O site CVS do Eclipse possui uma seção de Perguntas Frequentes que o ajudará com os problemas mais comuns que você pode enfrentar.

Para ajudar na instalação e configuração de um servidor CVS, consulte o site wiki do CVS em http://ximbiot.com/cvs. Procurar na Web por “como instalar um servidor cvs” também o levará a um host dos sites que descreverão em detalhes como instalar e configurar um repositório do CVS em vários sistemas operacionais e plataformas.

Compartilhando um Projeto

Você pode compartilhar um projeto com outros usuários usando CVS.

Para compartilhar um projeto, você escolhe a opção **Equipe** > **Compartilhar Projeto...** no menu suspenso em um projeto.



Isso abre outra tela, na qual é possível especificar parâmetros para o repositório de controle de origem.

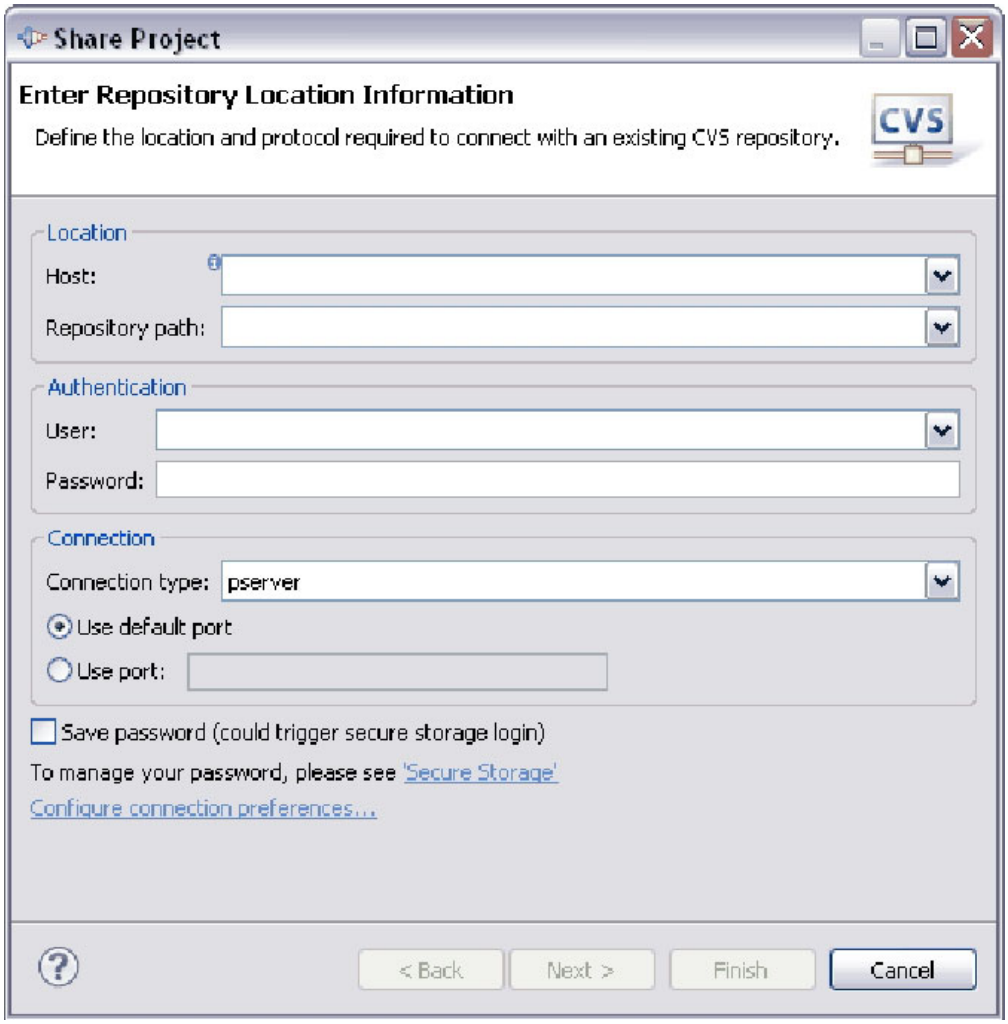
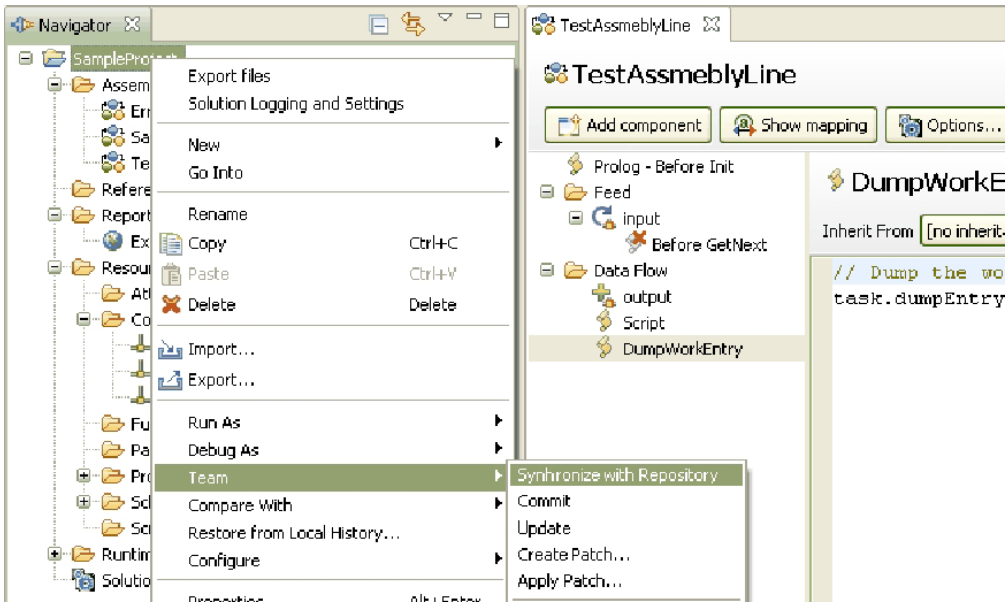


Figura 96. Janela Compartilhar Projeto do CVS

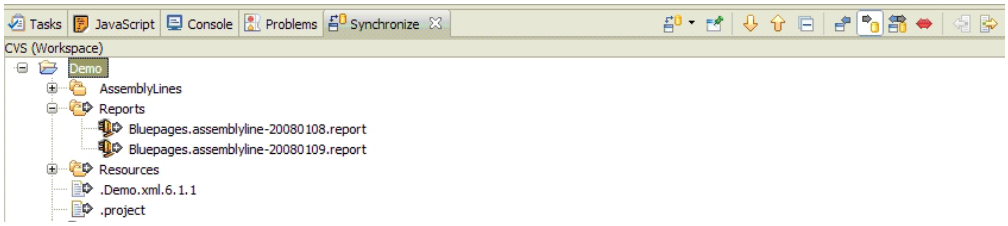
Conclua o assistente para compartilhar o projeto com o servidor CVS.

Nota: Somente os arquivos reais no projeto, e a estrutura de diretórios que os contém, podem ser controlados adequadamente pelo repositório. Os diretórios vazios não são considerados.

Quando o projeto tiver sido compartilhado, você poderá sincronizar com o repositório para confirmar suas próprias mudanças, assim como receber mudanças feitas por outras pessoas.



Selecione **Equipe** > **Sincronizar com Repositório** para abrir a visualização de sincronização:

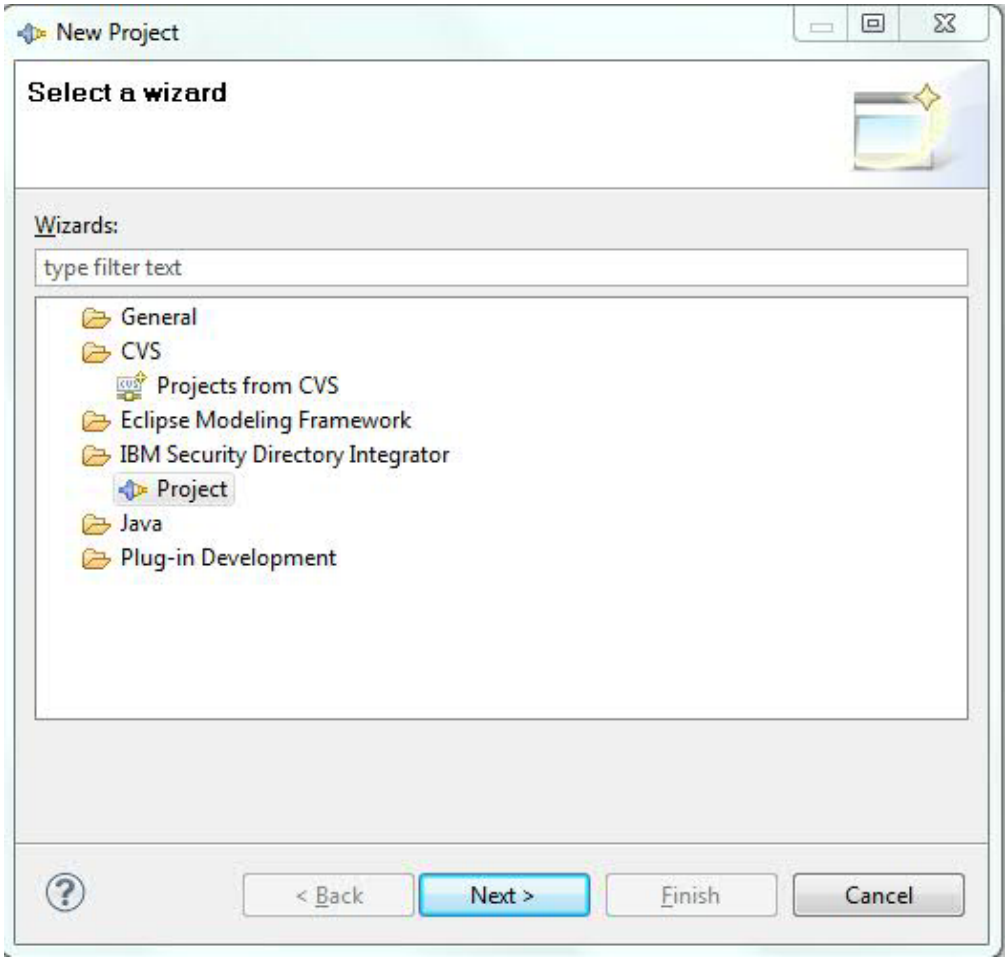


Essa visualização mostra quais arquivos precisam ser atualizados, assim como para quais arquivos há atualizações. Após confirmar as mudanças, o navegador mostrará informações adicionais para os arquivos.

Usando um Projeto Compartilhado

Se você desejar usar um projeto que alguém compartilhou, use o assistente de projeto CVS.

Selecione **Arquivo** > **Novo** > **Novo Projeto...** no menu principal e escolha o assistente de projeto CVS.



Complete o assistente para recuperar o projeto em sua área de trabalho.

Visualização Problemas

Quando você salva um componente, o construtor de projetos do IBM Security Directory Integrator atualizará o arquivo de configuração de tempo de execução para refletir as alterações feitas.

O construtor de projetos executará também uma verificação da validação nos avisos e erros dos componentes e relatórios da visualização Problemas. Na visualização Problemas você verá avisos como estes:

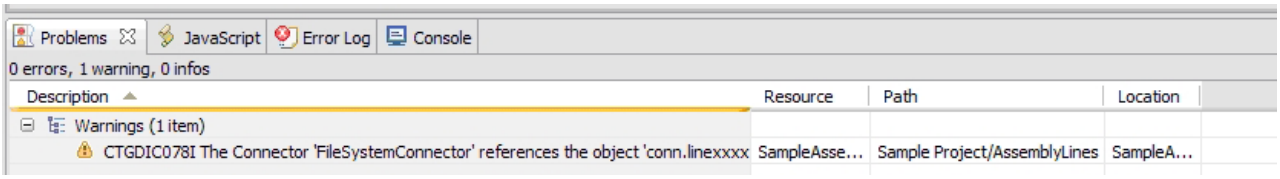


Figura 97. Janela da Visualização Problemas

Quando você clicar duas vezes em uma linha nessa visualização, ela abrirá o local em que o problema estava localizado.

Os problemas que você pode esperar ver na visualização Problemas são os seguintes itens:

- Referência a um item de esquema indefinido

  Se você usar construções como “conn.abc” e “abc”, não está definido no esquema que obterá esse aviso.
- Referência a um atributo de trabalho indefinido

  Se você usar construções como “work.abc” e “abc”, não se sabe se você receberá esse aviso.
- Erros sintáticos em scripts
- AssemblyLines com mais de um conector de modo de servidor.

Aprimoramentos de JavaScript

Em qualquer lugar que você edite scripts no Editor de Configuração, obterá um editor de texto, especificamente aprimorado para editar o código JavaScript.

Este editor fornece conclusão de código, sintaxe de cores, bem como marcação de erros sintáticos.

Alguns dos aprimoramentos são:

- “Conclusão de Código”
- “Coloração da Sintaxe” na página 173
- “Verificação de Sintaxe” na página 173
- “Avaliação Local” na página 173
- “Editores Externos” na página 174

Conclusão de Código

O editor JavaScript suporta conclusão de código.

À medida que você digita, o editor reage a determinados pressionamentos de teclas. O ponto (.) ativa a conclusão de código que abre um menu pop-up mostrando todos os métodos, campos e recursos relevantes específicos do IBM Security Directory Integrator. A conclusão de código pode também ser ativada manualmente pela combinação de teclas `Ctrl-<Espaço>`. A conclusão do código cobre os tipos de script JavaScript padrão (ou seja, sequência, número, etc.), assim como conclusão customizada para objetos específicos para o IBM Security Directory Integrator. Objetos como *conn* e*work* fornecem uma lista de atributos disponíveis, bem como campos e métodos do objeto.

A conclusão de código funciona contanto que o editor possa derivar o nome da classe Java na expressão:

O editor reagirá também a aspas simples ou duplas e a chaves ("{}"). Quando você digita uma aspa simples ou dupla, o editor insere automaticamente outra aspa e posiciona o circunflexo entre as duas. Isso é feito para facilitar a entrada de constantes da cadeia.

Digitar chaves indentará automaticamente para acomodar a indentação de blocos. Quando você digita uma abertura de chave e pressiona Enter, as guias serão inseridas e o circunflexo será posicionado na próxima linha com a indentação correta. Ou então, digitar um fechamento de chave cancelará a indentação da chave.

Mapeamento e Conclusão de Código

Quando você inclui atributos no CE, geraremos expressões simples baseadas no nome do atributo. Qualquer nome que contenha pontos ou outros caracteres que não sejam identificadores válidos de objetos de script, estará contido em ["nome do atributo"] (por exemplo, `conn["http.body"]`). Seja a Entrada hierárquica ou não irrelevante quando usamos essa notação desde que ela funcione nos dois casos. Atributos que são identificadores de javascript válidos são utilizados como (por exemplo, conn.cn). Se você tiver um esquema hierárquico, podemos gerar uma única expressão para o mapa. Se, por exemplo, você tiver uma hierarquia igual a a->b->c e mapear a parte "c", geramos ["a.b.c"] para referência do atributo.

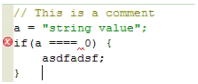
No editor de script você verá que a conclusão de código funciona de maneira semelhante. A conclusão de código mostrará as conclusões em um texto simples (por exemplo, http.body), mas quando você pressionar Enter para concluir a expressão, a mesma delimitação será utilizada para nomes de atributos que não sejam nomes de objetos de script válidos.

Coloração da Sintaxe

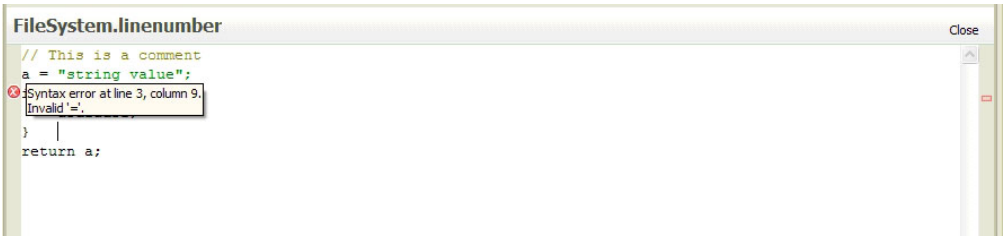
A coloração da sintaxe no editor é básica. O editor decora comentários e cadeias.

Verificação de Sintaxe

À medida que você digita, o editor executa uma verificação de sintaxe no script em segundo plano. Quando há erros no script, ele mostra o erro nas margens e o texto receberá um til vermelho para marcar onde o interpretador JavaScript encontrou o erro.



Quando você passa o mouse sobre a marca de erro na *régua* (na margem esquerda), vê a mensagem de erro em uma janela pop-up.

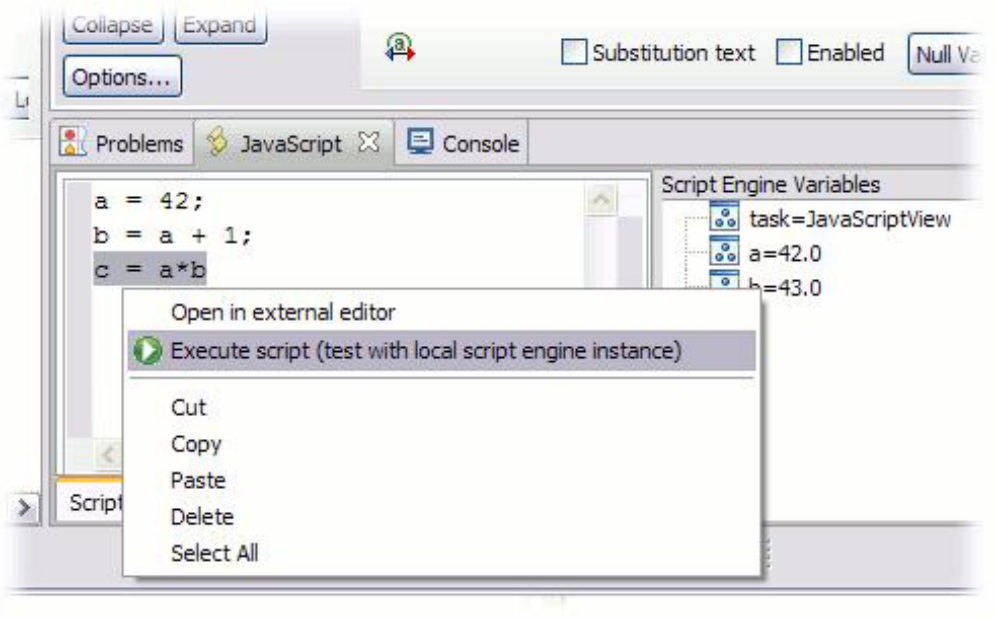


A régua à esquerda está sincronizada com a janela de texto e uma marca está diretamente relacionada à linha da janela de texto. Se não houver erros no texto que você vê na janela, não haverá marcas nesta régua também.

No entanto, a régua de visão geral à direita mostrará todos os erros do script inteiro, independentemente de onde o editor de texto está posicionado. Quando você passa o mouse sobre a marca da régua, obtém o mesmo pop-up de mensagem de erro obtido na régua à esquerda. Clicar no marcador reposiciona o editor na linha em que o problema foi identificado.

Avaliação Local

Quando você edita um script, pode clicar com o botão direito do mouse e escolher **Executar Script** no menu suspenso para fazer uma rápida avaliação do texto selecionado. Um ambiente de teste mais detalhado para scripts é a visualização JavaScript, na qual você pode executar scripts e examinar as variáveis do mecanismo de script em uma janela separada:



Editores Externos

Você pode editar o script em seu editor JavaScript favorito usando o menu de contexto.

Clique com o botão direito do mouse no campo de texto e escolha **Abrir no editor externo**. O editor é configurado na guia de preferências **Janela** > **Preferências** > **Security Directory Integrator**:

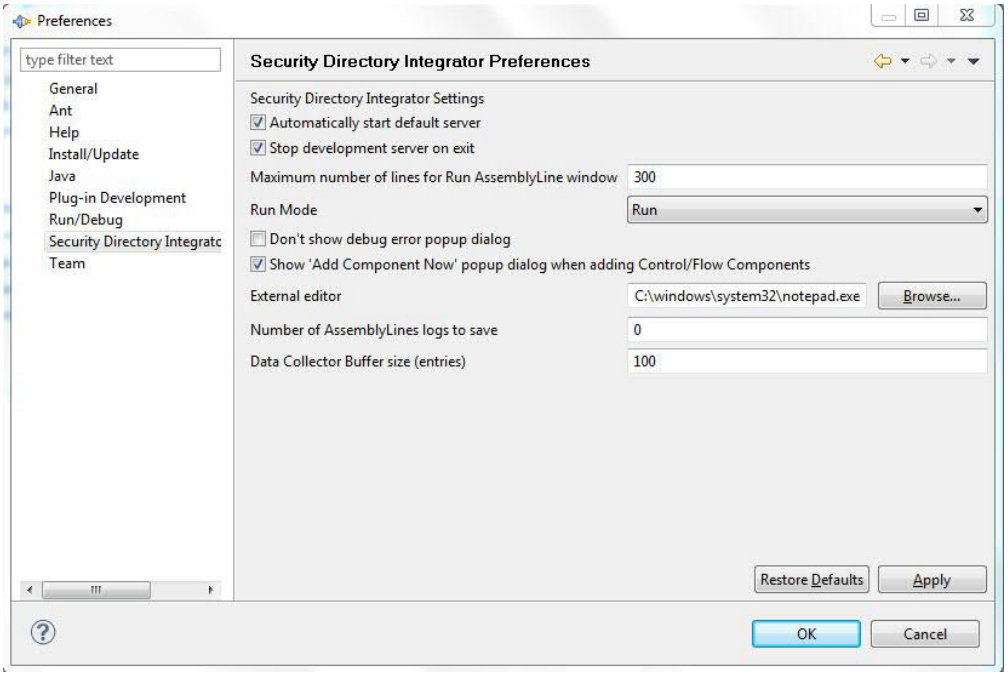
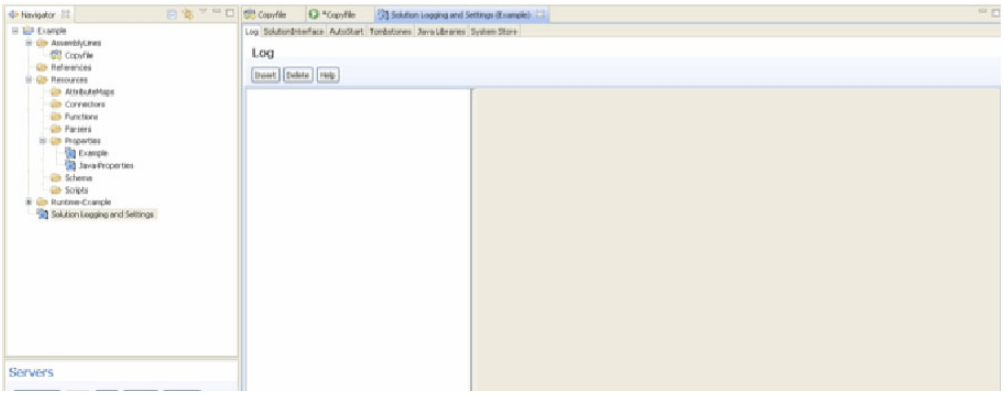


Figura 98. Janela Preferências do Editor de Configuração

Configurações e Criação de Log de Soluções

A janela Configurações e Criação de Log de Soluções permite editar áreas específicas da solução do projeto.

Você pode abrir a janela selecionando um projeto, ou um arquivo do projeto, e clicar duas vezes em **Configurações e Criação de Log de Soluções** na árvore do projeto.



Em Configurações e Criação de Log de Soluções, você pode modificar os seguintes itens:

- "Configurações de Armazenamento do Sistema"
- "Criação de Log" na página 177
- "Tombstones" na página 177
- "Bibliotecas Java" na página 178
- "Autoinicialização" na página 178
- "Configurações da Interface de Solução" na página 178

Configurações de Armazenamento do Sistema

As configurações de armazenamento do sistema para um projeto podem substituir o armazenamento de sistema padrão definido pelo servidor IBM Security Directory Integrator. Quando está ativado, o armazenamento de sistema configurado é utilizado pela configuração no lugar do armazenamento de sistema do Servidor.

As configurações de armazenamento do sistema podem ocorrer em dois lugares:

1. No nível do Projeto do IBM Security Directory Integrator em sua área de trabalho; e
2. No nível do servidor IBM Security Directory Integrator.

As configurações no nível do Projeto têm precedência sobre as configurações no nível do Servidor, ou seja, se você tiver definido um armazenamento de sistema especificamente para seu projeto, esse armazenamento de sistema será utilizado durante a execução dos seus componentes no Editor de Configuração; se você não tiver definido nenhuma configuração de armazenamento de sistema no projeto, o armazenamento de sistema do Servidor que você utiliza para executar seus componentes será utilizado.

Configurações para Nível de Projeto

No menu suspenso do título, você pode escolher modelos predefinidos, além de carregar e salvar configurações de armazenamento do sistema em arquivos na área de trabalho. Observe que todas as tabelas (delta, propriedades e outros) serão armazenadas nesse banco de dados.

Os itens de menu como **Derby Embedded** entre outros são modelos predefinidos que você pode carregar no painel de configuração, sendo que depois é possível modificá-los de acordo com suas necessidades e, subsequentemente, atualizá-los na configuração. Também é possível **Carregar Modelo...** a partir do sistema de arquivos local.

As mudanças feitas nesse painel são salvas no arquivo de Configuração (quando o projeto é exportado) e são utilizadas por todos os AssemblyLines, apesar das configurações do servidor no qual elas são executadas.

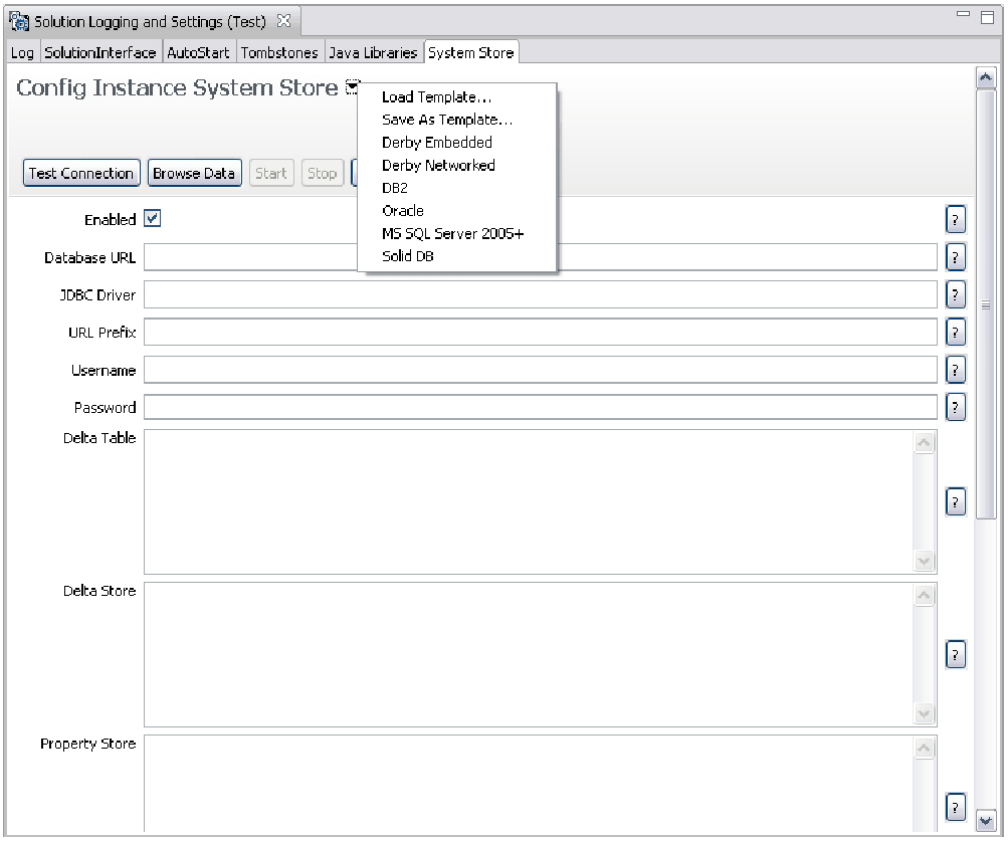


Figura 99. Definições do Armazenamento de Sistema de Configuração

Configurações para Nível de Servidor

A mesma tela aparece quando você seleciona **Editar Configurações de Armazenamento do Sistema** no menus suspenso em um servidor, na visualização Servidores. Isso atualizará e salvará as variáveis de configurações de armazenamento do sistema no arquivo de propriedades da solução. Uma reinicialização de servidor é necessária para ativar as novas configurações.

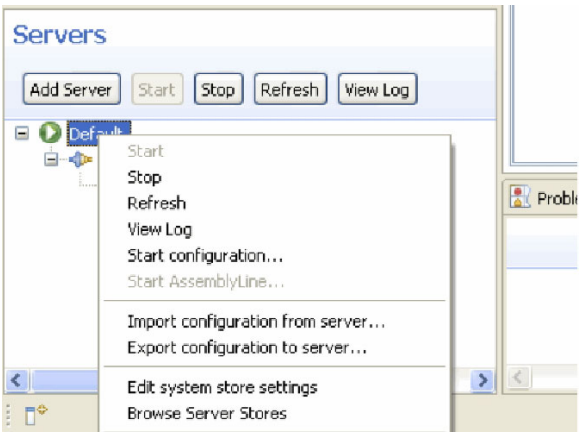
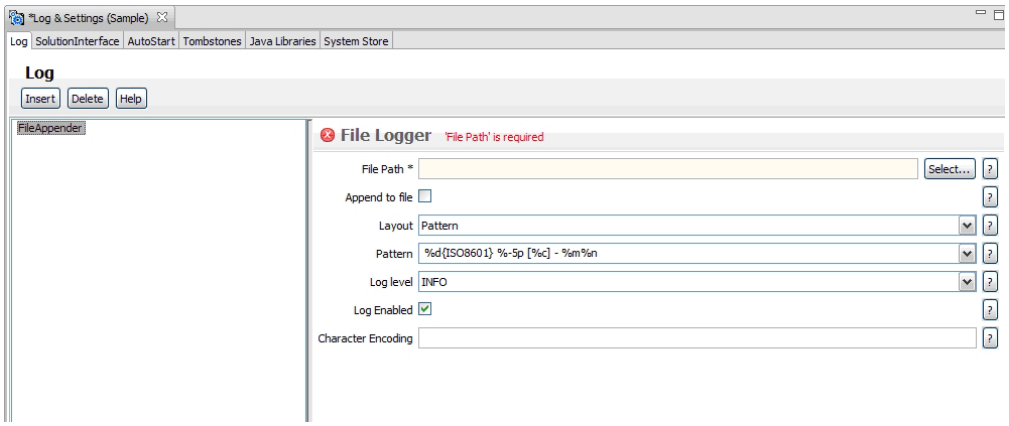


Figura 100. Menu de Contexto do Documento do Servidor

Criação de Log

A visualização **Criação de Log** mostra os criadores de log para a solução.

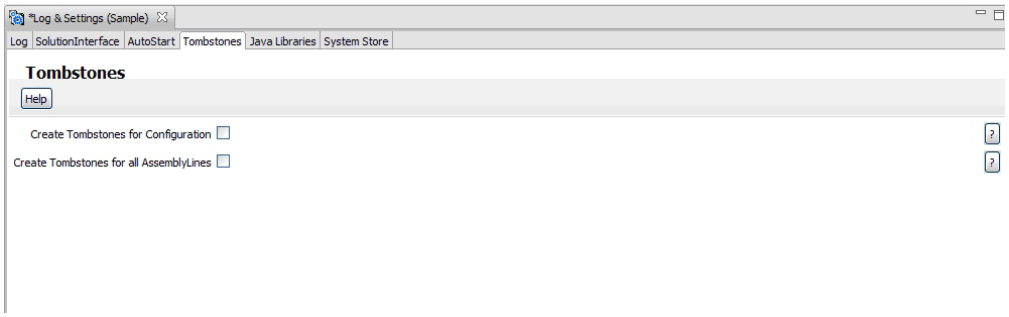
Você pode incluir e remover criadores de log, clicando em **Inserir** e **Excluir** na barra de título.



Consulte a seção "Criação de Log e Depuração" em *Instalando e administrando* para obter informações adicionais.

Tombstones

A configuração de Tombstones para um projeto está localizada na guia **Tombstones**.



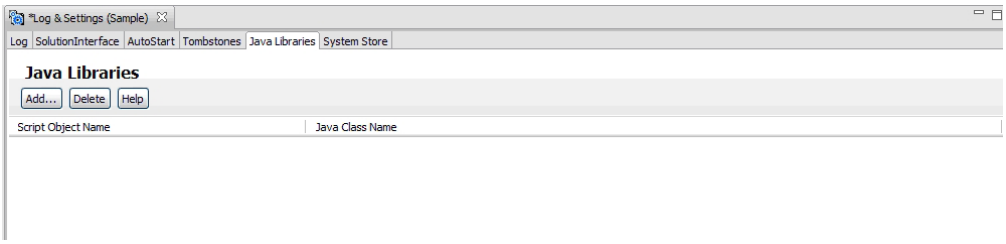
Um Tombstone é um registro de uma execução de um AssemblyLine, contendo algumas estatísticas, como número de entradas lidas por Agentes Iterativos, número de registros ignorados, número de registros atualizados e outros.

A seleção de **Criar Tombstones para Configuração** faz com que um Tombstone seja gerado toda vez que o arquivo de configurado derivado desse projeto, carregado em um Servidor IBM Security Directory Integrator, é finalizado.

A seleção de **Criar tombstones para todos os AssemblyLines** faz com que um Tombstone seja gerado toda vez que um AssemblyLine nesse projeto, executado em um Servidor, termina.

Bibliotecas Java

A guia **Bibliotecas Java** mostra as classes Java que são carregadas e definidas automaticamente em cada instância do mecanismo de script.



Autoinicialização

A guia **Autoinicialização** mostra uma lista na qual você pode inserir o nome de AssemblyLines que são iniciados automaticamente quando a instância de configuração é iniciada.

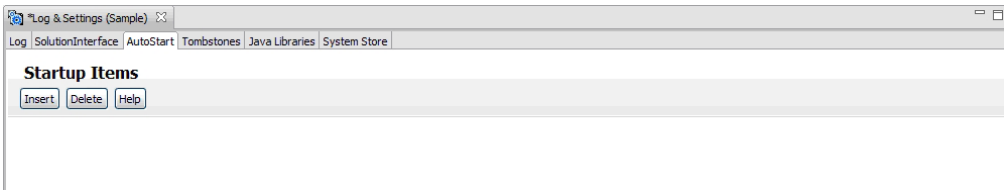


Figura 101. Configurações de Autoinicialização

Você pode incluir itens na lista clicando em **Inserir**; isso permite escolher a partir dos AssemblyLines existentes em sua área de trabalho.

Você pode remover os AssemblyLines na lista Itens de Inicialização, selecionando-os e clicando em **Excluir**.

Configurações da Interface de Solução

As configurações da interface de solução podem ser ativadas para fornecer informações adicionais sobre a configuração. Essas informações são geralmente usadas pela ferramenta Webadmin (AMC), mas podem ser usadas por outros clientes que possuem acesso à configuração da interface de solução.

Os dois primeiros campos neste painel são o nome da solução e o status ativado. O valor-padrão para o nome da solução é o próprio nome do projeto; se o nome da solução for deixado em branco, o arquivo de configuração exportado não conterá nenhum ID de Solução. A caixa de opção ativada determina se a configuração está

em uso ou não.

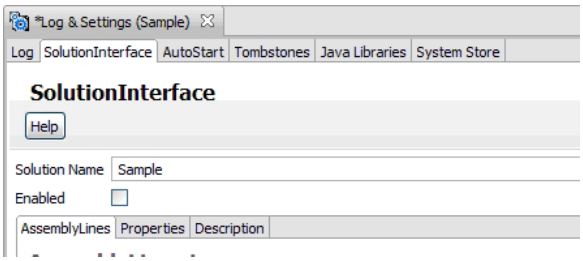


Figura 102. Configurações da Interface de Solução

As configurações da Interface de Solução são divididas em três seções, cada uma com sua guia correspondente:

- "AssemblyLines"
- "Propriedades" na página 180
- "Descrição" na página 180

AssemblyLines

Essa guia mostra todos os AssemblyLines no projeto e você pode marcar aqueles que devem ficar visíveis ao usuário para serem iniciados/parados.

O AssemblyLine de funcionamento é um AssemblyLine especial que é usado para relatar o funcionamento da configuração em execução. Esse AssemblyLine é simplesmente um que possa fornecer feedback customizado ao usuário quanto ao estado da configuração. O AssemblyLine de funcionamento é chamado e deve retornar dois campos em sua entrada de trabalho para relatar o status (`healthAL.result` e `healthAL.status`). Consulte a documentação do Action Manager (AM) (on-line no AMC, ou no *Instalando e administrando*) para obter informações adicionais sobre como esses campos são usados naquele contexto. O intervalo de sondagem especifica com que frequência o cliente (por exemplo, AMC ou AM) chamará o AssemblyLine de funcionamento.

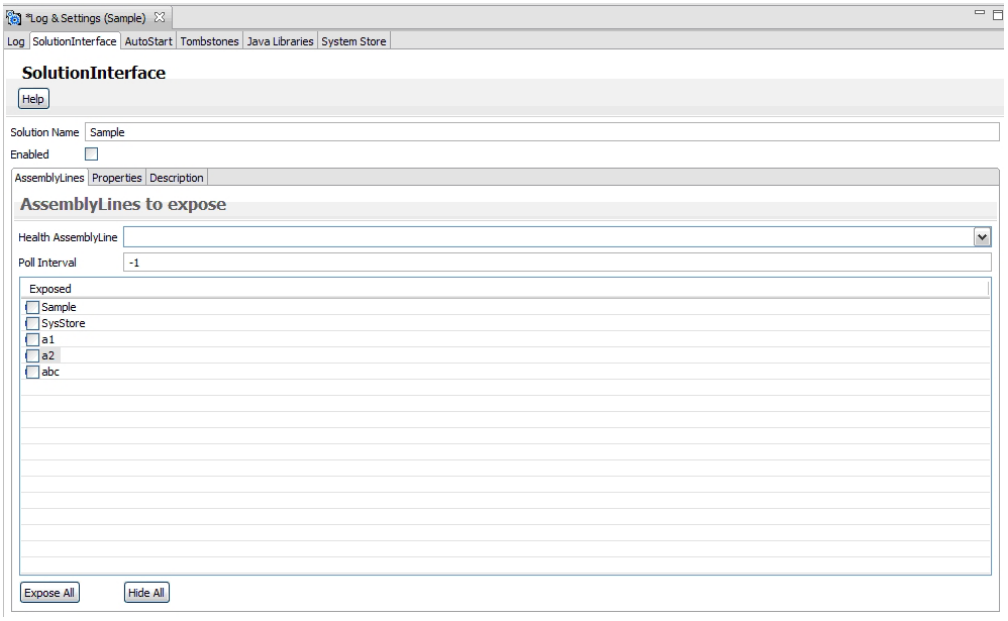


Figura 103. Configurações da Interface de Solução: AssemblyLines

Propriedades

Essa guia permite definir quais propriedades os usuários vêem e como ela é apresentada ao usuário.

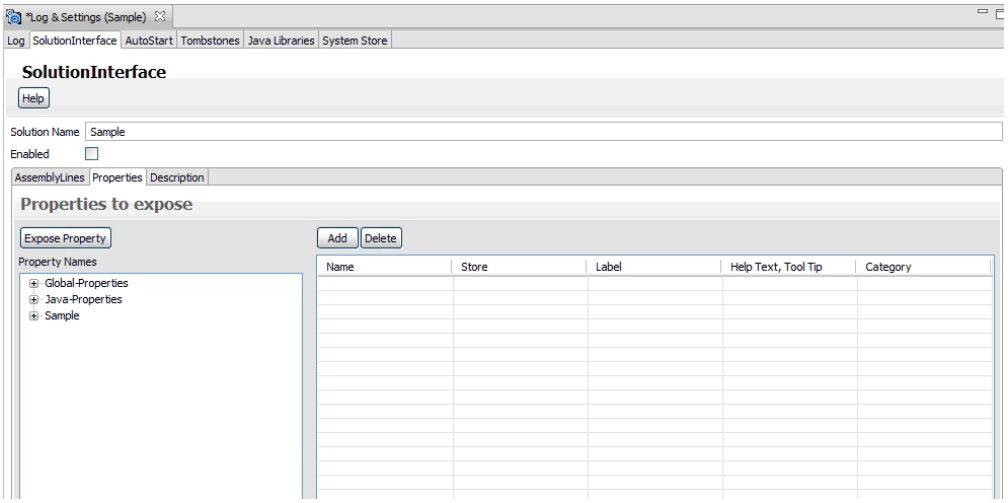


Figura 104. Configurações da Interface de Solução: Propriedades

Descrição

Essa guia permite inserir o texto que descreve a solução. Ela destina-se somente aos propósitos da documentação; não é usada de nenhuma outra maneira pelo IBM Security Directory Integrator.

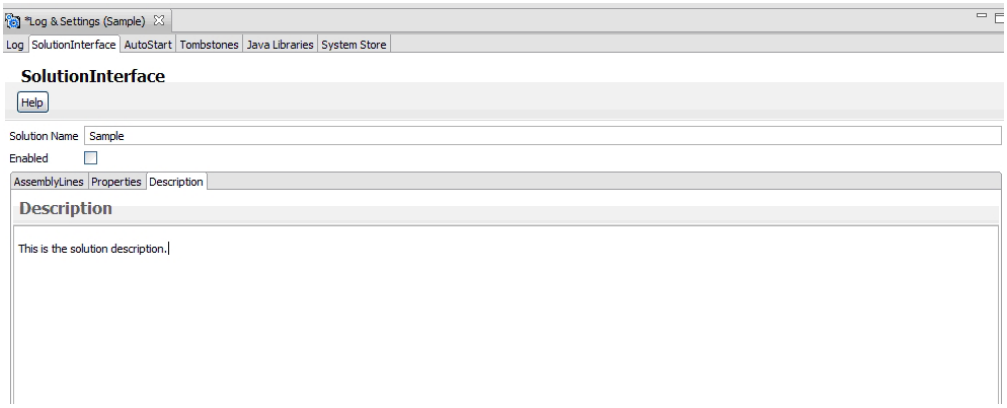


Figura 105. Configurações da Interface de Solução: Descrição

Propriedades do Servidor

Você pode editar propriedades para um projeto usando o editor de Propriedades.

Crie um novo arquivo de propriedades usando o assistente **Arquivo** > **Novo** > **Propriedades** e digite o nome de um armazenamento de propriedades.

No editor de propriedades, você pode trocar o conteúdo do armazenamento de propriedades usando os comandos **Download** e **Upload**:

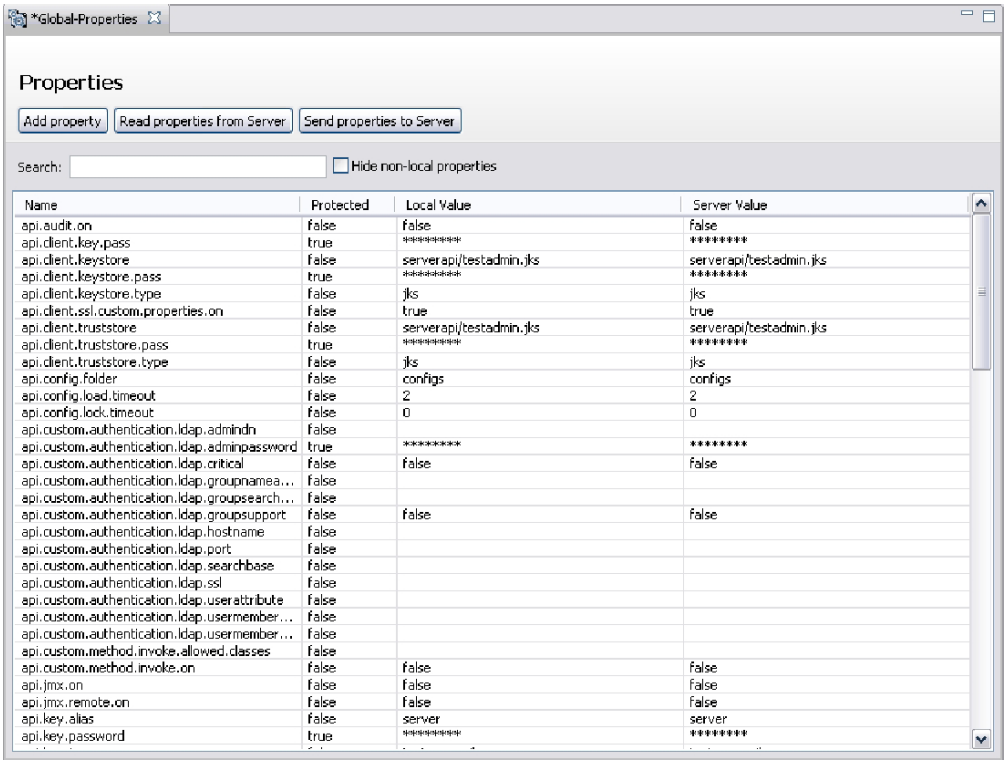


Figura 106. Janela Editor de Propriedades

Utilize **Download** para recuperar todas as propriedades do armazenamento de propriedades (por exemplo, o arquivo de propriedades designado a esse

armazenamento). Observe que esses valores são lidos e transmitidos para o CE a partir do IBM Security Directory Integrator Server atualmente selecionado para o projeto. Inversamente, o botão **Upload** atualiza o armazenamento de propriedade (via o servidor atual) com os valores no editor. Somente as propriedades que possuem um valor local são atualizadas.

Use o campo de texto **Procurar** para mostrar as propriedades que correspondem ao texto nesse campo. Marcar **Ocultar Propriedades Não-locais** faz com que o editor mostre somente as propriedades que possuem um valor local.

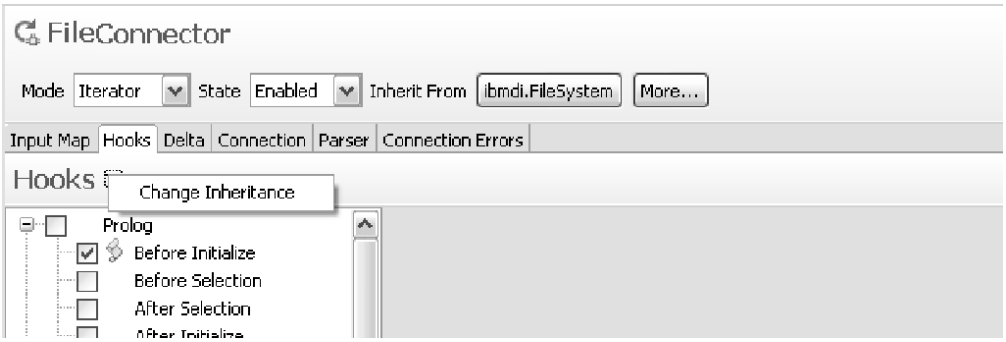
O construtor de projetos incluirá a configuração do armazenamento de propriedades no arquivo de configuração executável. Entretanto, os valores das propriedades neste documento são transferidas somente conforme necessário.

Nota: A ordem em que os armazéns de Propriedades são inicializados e acessados no Servidor IBM Security Directory Integrator é indefinida. Portanto, não é possível armazenar confiavelmente em um armazenamento de Propriedades quaisquer propriedades que definem os parâmetros de acesso (por exemplo, nomes de arquivos) de outros armazenamentos de Propriedades.

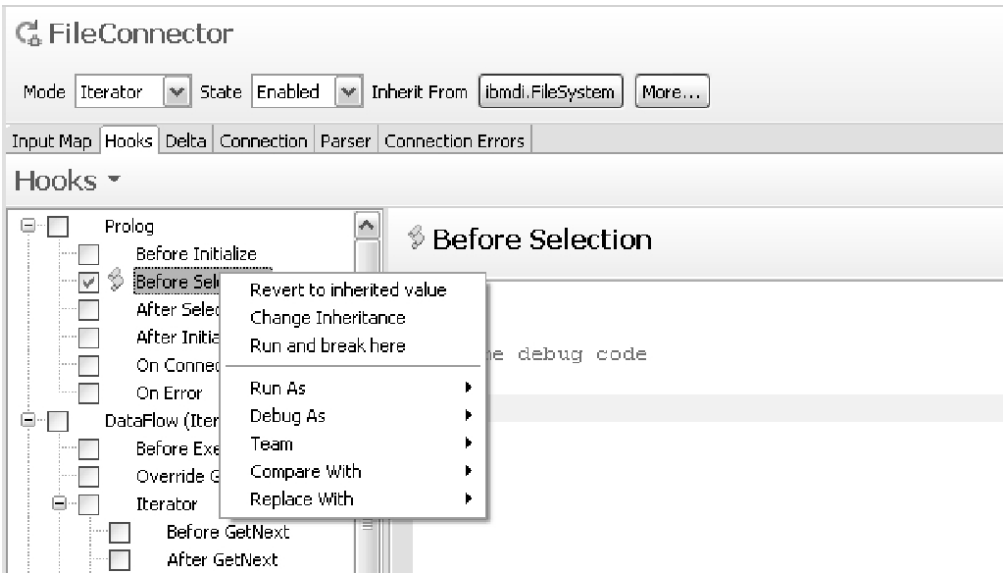
Herança

Os componentes podem herdar elementos de outros componentes arrastando e soltando um objeto na barra de título de um componente ou uma subseção do componente.

Nas subseções, há também uma opção de menu (**Alterar herança**) disponível na barra de título.



Nos ganchos e mapas de atributos, há uma seleção de herança separada em itens individuais em que você pode optar por herdar de um componente de script ou de função da área de trabalho.

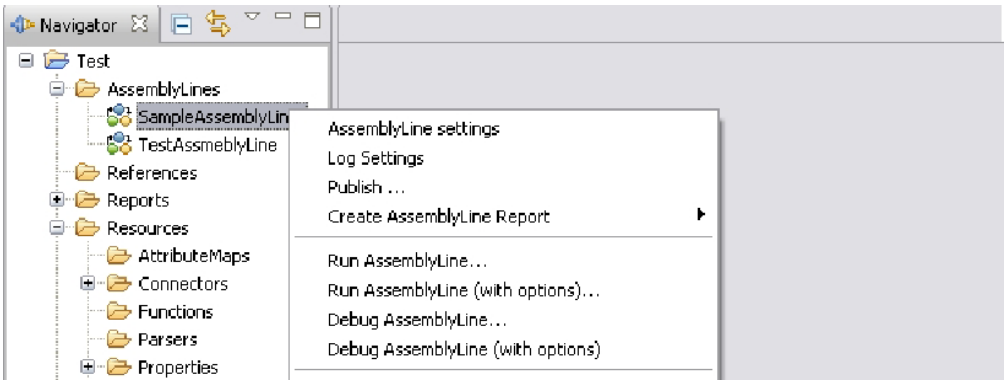


Quando um item de configuração substitui um item herdado, a interface com o usuário oferece uma maneira de reverter para o valor herdado por meio de um item de ação do menu suspenso **Reverter para o valor herdado**.

Ações e Ligações de Teclas

A CE contribui com várias ações para si própria, bem como a objetos do ambiente de trabalho. Essas ações executam operações específicas em objetos específicos.

Por exemplo, **Executar Relatório do AssemblyLine** é uma ação que é executada em todos os arquivos com uma extensão igual a `.assemblyline`. Quando você clica com o botão direito do mouse em um AssemblyLine do navegador, o menu suspenso incluirá esse comando bem como todas as outras contribuições nesse tipo de objeto.



Essas ações tem também uma definição de comando associada. Uma definição de comando permite que o usuário defina o atalho de teclado para um comando. Isso é feito no painel **Janela** > **Preferências** > **Teclas**:

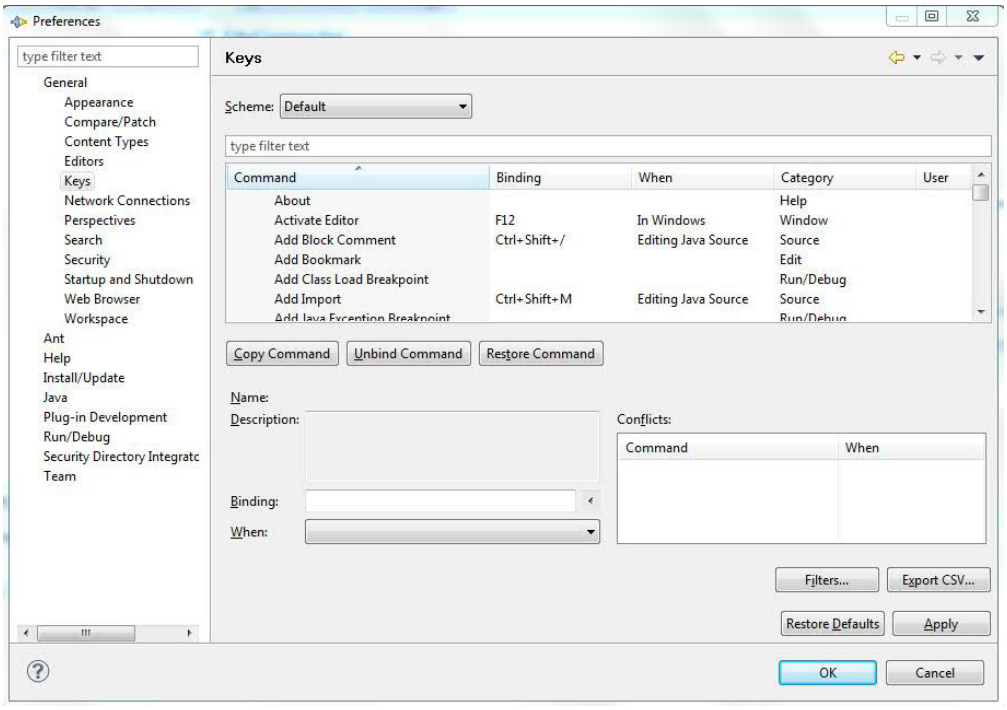
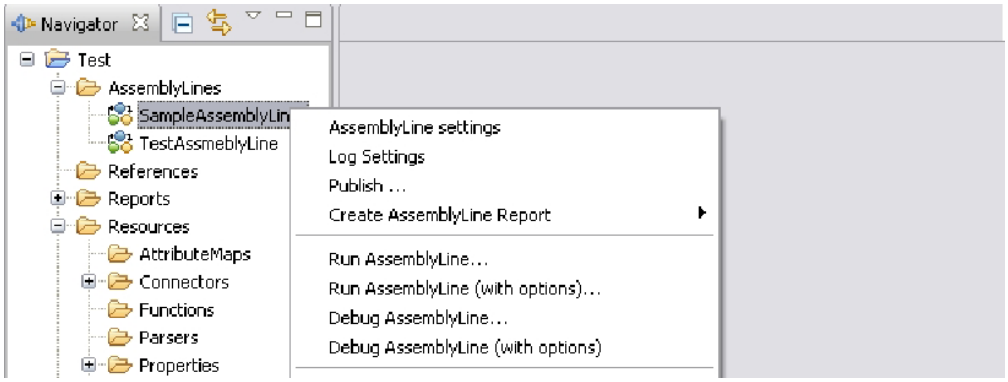


Figura 107. Janela Designações de Teclas

A figura acima mostra como as designações de teclas são feitas na interface com o usuário. Nesse exemplo, o comando Executar relatório foi designado `Alt+Shift+I` como seu atalho. Ao abrir o menu novamente no AssemblyLine, você verá isso refletido no menu.



É possível obter uma lista de todos os comandos específicos do IBM Security Directory Integrator inserindo o texto `integrador de diretórios de segurança` no campo de procura, que pode ser localizado abaixo do seletor de **Esquema**.

Capítulo 4. Recursos de Depuração no IBM Security Directory Integrator

Depois de criada a sua solução do IBM Security Directory Integrator, geralmente usando o Editor de Configuração (CE), há várias maneiras de colocá-la no ritmo e testá-la. Alguns dos recursos de depuração que o IBM Security Directory Integrator oferece estão disponíveis de dentro do CE e alguns baseiam-se em scripts que estão disponíveis no diretório de instalação do IBM Security Directory Integrator.

Os recursos de depuração no IBM Security Directory Integrator são Sandbox, Modo de Simulação do AssemblyLine e stepper e depurador.

O stepper e o depurador fazem parte do Editor de Configuração; consulte "O Stepper e o Depurador" na página 155.

Sandbox

O IBM Security Directory Integrator inclui um recurso Sandbox que permite que você registre a operação de um ou mais Conectores em um AssemblyLine para responder posteriormente sem a necessidade de as origens de dados estarem disponíveis.

O recurso Sandbox utiliza Armazenamento do Sistema. Consulte Capítulo 6, "Armazenamento do Sistema", na página 203 para obter informações adicionais.

Registrar um componente significa que o AssemblyLine interceptará cada chamada da instância do conector usada pelo componente. A gravação de um componente conector, por exemplo, gravará todas as chamadas para seus métodos de instância do conector, como `selectEntries`, `getNextEntry` e outros. O resultado de cada uma dessas chamadas é registrado no banco de dados de sandbox configurado. As informações registradas são o valor de retorno ou a exceção lançada pelo conector.

Para executar uma sessão de teste, crie um AssemblyLine e inclua um conector do sistema de arquivos que leia dados a partir de um arquivo. Inclua um componente de script que efetue dumps da entrada de trabalho. Em seguida, marque o conector do sistema de arquivos deste painel e selecione **Executar/Registrar** no menu do botão Executar. Depois de ter feito isso, você poderá mover o arquivo que acabou de ler e executar o AssemblyLine no modo Reprodução. Você deverá ver a mesma saída de log, mesmo se o conector do arquivo normalmente for interrompido, uma vez que o arquivo não está mais acessível.

Esse recurso pode ser muito útil ao fornecer materiais de suporte. Geralmente, o momento de reproduzir o ambiente de um AssemblyLine e o estado de origens de dados para reproduzir uma condição podem ser amplamente abrangentes. Com um banco de dados de sandbox com uma sessão registrada, uma pessoa do suporte pode executar o AssemblyLine sem precisar acessar todos os armazenamentos de dados que o AssemblyLine exige. Além disso, a configuração do AssemblyLine pode ser modificada para imprimir mais informações, caso isso seja necessário. A única alteração que não pode ser feita na configuração do AssemblyLine é fazer chamadas adicionais ou solicitar novamente as chamadas para componentes registrados. Isso causaria um erro durante a reprodução, uma vez que as chamadas para os conectores não corresponderiam à próxima chamada esperada para o conector.

Para que possa gravar ou reproduzir um AssemblyLine, você deve primeiro informar ao IBM Security Directory Integrator onde armazenar os dados de registro do AssemblyLine. Isso é feito na janela Sandbox, que você pode abrir selecionando **Configurações do AssemblyLine...** > **Configurações de Sandbox** no Editor do AssemblyLine. Na parte superior dessa janela, há um campo chamado **Banco de Dados** no qual você pode digitar o caminho do diretório a ser usado pelo sistema.

O recurso Sandbox não é suportado nos AssemblyLines que contêm um Conector no modo Servidor, ou um Conector de Repetidor com Delta ativado. O servidor interromperá a execução do AssemblyLine se ele estiver descoberto.

Gravando a Entrada do AssemblyLine

Registrar um componente significa que o AssemblyLine interceptará cada chamada da instância do conector usada pelo componente. A gravação de um componente conector, por exemplo, gravará todas as chamadas para seus métodos de instância do conector, como `selectEntries`, `getNextEntry` e outros. O resultado de cada uma dessas chamadas é gravado no banco de dados do Sandbox configurado. As informações registradas são o valor de retorno ou a exceção lançada pelo conector.

Para executar uma sessão de teste, crie um AssemblyLine e inclua um conector do sistema de arquivos que leia dados a partir de um arquivo. Inclua um componente de script que efetue dumps da entrada de trabalho. Em seguida, verifique o conector do sistema de arquivos nesse painel e selecione **Executar/Gravar** no menu do botão **Executar**. Depois de pronto, você poderá mover o arquivo que acabou de ser lido e executar o AssemblyLine no modo Reprodução. Você deverá ver a mesma saída de log, mesmo se o conector do arquivo normalmente for interrompido, uma vez que o arquivo não está mais acessível.

Reprodução de Sandbox de Gravações do AssemblyLine

Quando um AssemblyLine está no modo Sandbox, todos os Conectores configurados para reprodução são considerados como no *modo virtual*. Isso significa que suas operações de Interface de Conector (por exemplo, `getNext()`, `findEntry()`) não são realmente chamadas. Em vez disso, essas operações são simuladas durante a reprodução.

Para executar um AssemblyLine no Modo de Reprodução de Sandbox, você deve selecionar os Conectores a serem executados no modo virtual, correspondendo a caixa de opção **Reprodução Ativada** na janela de configurações de **Sandbox** do AssemblyLine.

Nota: Nem todos os Conectores gravados precisam ser ativados para reprodução. Você pode ativá-los para acessar origens de dados ativos, embora isso possa afetar os resultados da operação de reprodução.

Para executar um AssemblyLine a partir da linha de comandos, inicie o servidor com a opção **-q2**. Um AssemblyLine no modo Sandbox é executado com entrada (incluindo sua Entrada de Trabalho Inicial) proveniente de um conjunto de dados gravados. Por exemplo, se você tiver um Conector Java Messaging Service (JMS) em seu AssemblyLine no modo Sandbox, o Conector JMS recuperará a entrada dos dados gravados anteriormente e nunca será realmente inicializado.

Ao registrar um AssemblyLine, o servidor cria um banco de dados Derby no diretório do **Banco de Dados** especificado, usando o nome do AssemblyLine como o nome do banco de dados. Esse banco de dados contém tabelas para cada

Conector no AssemblyLine. Um AssemblyLine no modo Sandbox pode ter um ou mais de seus Conectores virtuais substituídos, renomeando o Conector gravado e, em seguida, incluindo um novo com seu nome original.

Modo de Simulação do AssemblyLine

Os AssemblyLines podem ser depurados sem realmente trocar dados com os sistemas conectados usando o Modo de Simulação do AssemblyLine. Quando um AssemblyLine for iniciado neste modo, todos os Componentes do AssemblyLine que potencialmente possam causar mudanças nos sistemas de destino serão ignorados.

Isso significa que o AL é executado normalmente, mas os Conectores no Modo Atualizar, Delta, Excluir e AddOnly não executam a operação real.

Nota: Um AssemblyLine executado dessa maneira pode somente assemelhar-se ao que ocorreria no modo normal; em muitos casos, o simples fato de que um sistema conectado não recebe atualizações no Modo de Simulação pode fazer com que sua lógica de negócios se comporte diferentemente, anulando a utilidade da simulação.

O estado da simulação não deve ser confundido com o estado de um componente. O estado do componente possui prioridade mais alta que o estado de simulação do componente.

O estado do componente possui dois valores – Ativado ou Desativado. Se o componente estiver no estado Ativado, ele inicializará e seu estado de simulação será verificado quando a operação apropriada for chamada. Quando o estado estiver configurado como Desativado, nenhuma verificação quanto ao estado de simulação será feita, pois nenhuma operação será chamada, e a inicialização do componente não será chamada.

Os Conectores e os FCs possuem um outro estado chamado Passivo. Quando seus estados estiverem configurados como Passivo, eles somente serão inicializados. Suas operações podem ser executadas somente por meio de um script do usuário. Quando suas operações específicas forem chamadas, o estado de simulação será verificado.

Há várias maneiras em que um AssemblyLine pode ser iniciado no Modo de Simulação:

A partir do Configuration Editor:
Uma caixa de opção no menu suspenso do modo Executar está disponível para ativar e desativar a simulação para o AL. Você deve escolher o modo de execução desejado Executar com Opções no menu suspenso e marcar a caixa de opção **Modo de Simulação** na caixa de diálogo pop-up Opções a fim de fazer a simulação do AL durante a execução. O estado padrão dessa caixa de opção é desmarcado.

Usando o comando de inicialização do Servidor `ibmdisrv`:
Esse comando reconhece o comutador `–M` e iniciará o AL com a simulação ativada se esse comutador for fornecido.

Usando a API:
Para que um AL seja executado no Modo de Simulação, você deve configurar a propriedade `AssemblyLine.TCB_SIMULATE_MODE` como true no objeto TCB. Esse objeto deverá então ser fornecido para o método

`startAssemblyLine(String, TaskCallBlock)`. Se nenhuma propriedade for configurada, então, por padrão, seu valor será considerado como false.

Para executar um AssemblyLine no Modo de Simulação, uma nova configuração será criada. Ela será uma filha da Configuração do AssemblyLine. Esse objeto de configuração contém parâmetros que configuram a conexão com um AL que será usado como um Proxy AssemblyLine (ProxyAL). O objeto SimulationConfig possui um método que cria ou atualiza um modelo de um ProxyAL com base nos estados dos componentes do AssemblyLine que está sendo simulado. O objeto SimulateConfig também contém todos os ganchos definidos para os Componentes que estão em um estado de simulação Com Script. O nome de cada gancho é igual ao nome do componente no estado de simulação Com Script. Ele contém também o estado de simulação para cada componente.

Os AssemblyLines a serem executados no Modo de Simulação podem ser configurados com mais detalhes; as configurações relevantes podem ser configuradas selecionando **Configurações do AssemblyLine** > **Configurações de Simulação** na janela Editor de AssemblyLine.

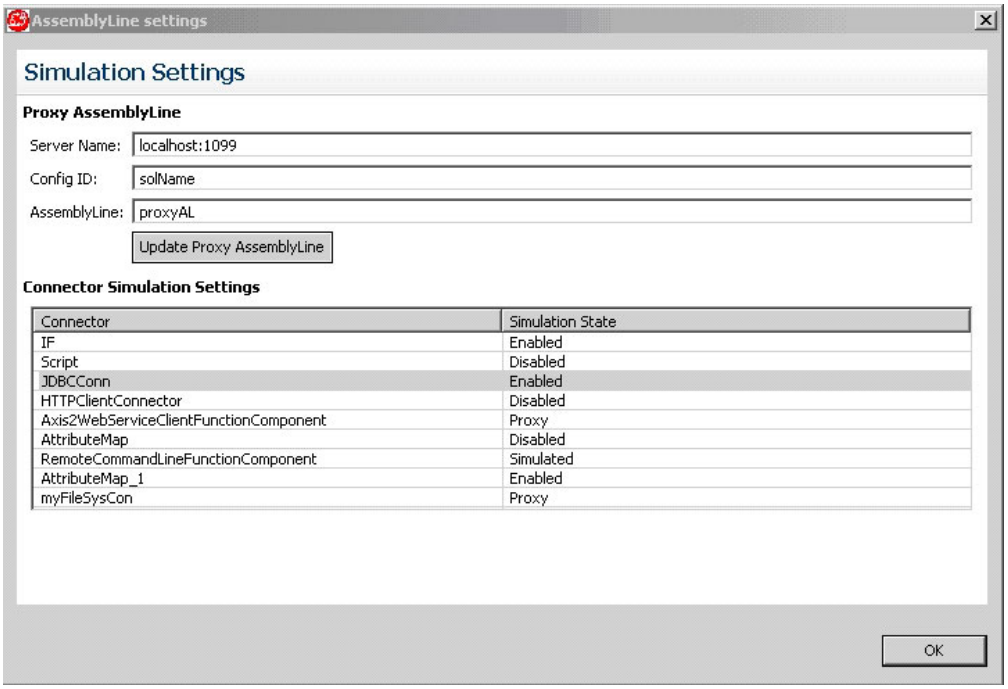


Figura 108. Janela Configurações de Simulação

A configuração dos estados de simulação para cada componente é feita usando a caixa de diálogo Configurações de Simulação. Essa caixa de diálogo configura o ProxyAL para ser usado pelos componentes cujo estado de simulação esteja configurado como *ProxyAL*. Ao clicar em **Atualizar AssemblyLine de Proxy**, o Config Editor criará um novo ProxyAL no projeto atual ou atualizará um AssemblyLine de Proxy existente. O ProxyAL criado ou atualizado será fornecido como um modelo e sua estrutura será baseada na configuração feita na caixa de diálogo Configurações de Simulação. Observe que nesse processo, somente o nome do AssemblyLine de Proxy é considerado. O Nome do Servidor e o ID de Configuração são considerados durante a execução do AssemblyLine Simulado. Se já houver um AssemblyLine existente com o nome especificado na caixa de diálogo, somente novas ramificações serão incluídas e nenhuma ramificação antiga

será modificada ou removida. Isso ocorre porque algumas delas podem conter uma configuração específica do usuário. Os componentes individuais no AssemblyLine podem ser configurados para um dos seguintes estados:

Ativado
Isso é o equivalente a executar esse componente no AL no modo normal (ou seja, o componente é executado tal como seria executado normalmente).

Desativado
Isso é equivalente a desativar o componente (ou seja, nenhuma operação, gancho ou qualquer outra coisa, exceto a inicialização, é executada para o componente).

Simulado
Geralmente, as estatísticas para todos os componentes descritos a seguir conterão informações para a operação que teria sido concluída se o AL não estivesse simulando. Como nenhuma operação crítica é feita durante a simulação, não há uma maneira possível de prever qual seria o resultado da execução de uma operação crítica (sucesso ou erro), portanto, as estatísticas determinarão que a operação foi concluída com êxito.

- Conectores no modo AddOnly: executados normalmente, somente a chamada potencialmente insegura para o método `connector.putEntry()` e a chamada para o gancho `override_add` são ignoradas.
- Conectores no modo Atualizar: executados normalmente, somente a chamada potencialmente insegura para o método `connector.modEntry()` e a chamada para o gancho `override_modify` são ignoradas.
- Conectores no modo Excluir: executados normalmente, somente a chamada potencialmente insegura para o método `connector.deleteEntry()` e a chamada para o gancho `override_delete` são ignoradas.
- Conectores no modo Delta: executados normalmente, mas devido ao fato de que o conector depende das chamadas para os métodos descritos acima, ele será simulado quando os métodos e ganchos acima forem ignorados.
- Conectores no modo Agente Iterativo com tags Delta: executados normalmente; para esses componentes, a mudança é semelhante aos Conectores descritos acima, ou seja, as chamadas para os métodos potencialmente inseguros para a classe BTree (`putEntry`, `modEntry`, `deleteEntry`) são ignoradas. Para o CDDeltaTaskComponent, o estado de confirmação é substituído para desativar a confirmação (“Nenhuma confirmação automática”) mas a confirmação ainda será possível para usuários que chamarem explicitamente o método `CSDeltaTaskCoponent#commitDeltaState()`.
- Function Components (FCs): executados normalmente, mas a chamada dos ganchos “`before_functioncall`”, “`after_functioncall`” e “`no_reply`” e também a chamada para o método `function.perform()` são desativadas. A chamada desses ganchos é desativada porque eles estão associados a um objeto retornado do método de execução que também está desativado. A única coisa que será feita é uma mudança nas estatísticas que indicarão que o FC executou com êxito essa operação.

  Além disso, os Mapas de Entrada e Saída serão executados mas a entrada real antes do InputMap será vazia (isso pode levar ao lançamento de uma exceção; é melhor desativar cada mapeamento de atributo simples do InputMap que depende de um atributo retornado do Function Component ou incluir uma verificação de validade do atributo

recuperado no Mapa de Atributo avançado; alternativamente, você pode usar o mecanismo NullBehaviour para substituir potenciais erros).
- Os conectores em qualquer ou modo são executados da maneira usual.

Proxy Os conectores nesse estado de simulação iniciarão um outro AL (especificado na guia Simulação) que substituirá a execução potencialmente insegura da operação. Esse AL externo é compartilhado entre todos os Componentes nesse modo de simulação. Ele será executado com uma operação diferente cujo nome corresponde ao nome do Componente que está sendo simulado. Consulte "Fluxo de Trabalho do AssemblyLine do Proxy" na página 191.

Com Script
Um script definido pelo usuário substituirá a operação potencialmente insegura. Cada componente nesse modo de simulação possui seu próprio gancho, diferente do estado Proxy, em que um AL é compartilhado entre os componentes. Consulte "Fluxo de Trabalho do Script de Simulação" na página 192.

É possível verificar se o AL está em simulação e também ativar e desativar a simulação usando os seguintes métodos:
- `boolean AssemblyLine#isSimulating()` Usado da seguinte forma a partir de um gancho: task.isSimulating();
- `void AssemblyLine#setSimulating(boolean)` Usado da seguinte forma a partir de um gancho: task.setSimulating(true);

É possível verificar o estado de cada componente e configurá-lo dinamicamente usando os seguintes métodos:
- `String AssemblyLineComponent.getSimulatingState()` Usado da seguinte forma a partir de um gancho: ConnectorName.getSimulatingState();
- `void AssemblyLineComponent.getSimulatingState(String)` Usado da seguinte forma a partir de um gancho: ConnectorName.setSimulatingState("Proxy");

Nota: Somente Conectores e FCs possuem o conjunto completo de estados de simulação conforme descrito acima, os componentes restantes (e Conectores no modo Servidor) possuem somente os estados Ativado e Desativado.

O IBM Security Directory Integrator considera alguns FCs seguros e o estado de simulação padrão deles será Ativado, ou seja, por padrão, eles não serão simulados e serão executados da maneira usual; estes são todos os FCs que não podem alterar um sistema de destino porque simplesmente não se conectam a nenhum. A lista de FCs seguros é a seguinte:
- FC CBE
- FC JavaToXML
- FC XMLToJava
- FC SDOToXML
- FC XMLToSDO
- Analisador do FC
- FC MemQueue
- FC JavaToSOAP
- FC SOAPToJava
- FC WrapSOAP

Atenção: Você ainda pode causar mudanças nos sistemas de dados subjacentes por codificação explícita; isto está fora do escopo do Modo de Simulação.

Quaisquer FCs restantes não listados aqui serão considerados potencialmente inseguros e seus estados de simulação padrão será configurados como Simulado.

Fluxo de Trabalho do AssemblyLine do Proxy

No contexto do Modo de Simulação, a chamada para um AL do proxy funciona de modo semelhante aquela que você usaria no Conector do AssemblyLine para especificar um AL de sua escolha; no entanto, algumas diferenças significativas são observadas.

A chamada do gancho de substituição da operação específica é desativada quando o componente está no estado de simulação do proxy.

A tabela abaixo mostra os métodos que são chamados quando o Conector está em um desses Modos ou quando o componente for um componente de Função.

Tabela 10. Solicitação de Método de Acordo com o Modo

Modo	Método
Conector: AddOnly	putEntry
Conector: Atualizar	findEntry, modEntry, putEntry
Conector: Excluir	findEntry, deleteEntry
Conector: Delta	findEntry, modEntry, putEntry, deleteEntry
Conector: Agente Iterativo	selectEntries, getNextEntry
Conector: CallReply	queryReply
Conector: Consultar	findEntry
Componente de Função	executar

Quando chega a hora de chamar um método específico e o componente está no estado de simulação Proxy, então a chamada para esse método será delegada para o AL do proxy. Quando o AL do proxy for iniciado, uma op-entry será transmitida para ele com os seguintes atributos:

- $operation – esse atributo contém o nome da operação a ser executada. Quando o AL do proxy for chamado, em vez de um método específico do componente, então o valor desse atributo será o mesmo que o nome do componente.
- $method – esse atributo contém o nome do método que seria normalmente chamado, mas como o componente está no estado de simulação Proxy e Componentes em alguns modos (por exemplo, Atualizar) executam vários métodos antes de realmente fazer a modificação (ou seja, findEntry) então o AL do proxy deverá reconhecer qual método implementar. Esse atributo $method apenas informa ao AL do proxy qual método é executado para que o AL do proxy possa manipular a operação corretamente.
- search – esse atributo estará disponível quando o atributo $method for findEntry. Seu valor é um objeto do tipo SearchCriteria e representa os Critérios de Procura definidos pelo usuário. Por exemplo, se o componente estiver em um estado de simulação Proxy e seu modo for Atualizar, então um Critério de Procura deverá ser definido para que você possa atualizar a entrada correta do sistema de destino. Como o estado de simulação é Proxy, a operação de consulta real que ocorre antes da operação de modificação delega sua execução ao AL do proxy. Em seguida, o AL do proxy usa esses Critérios de Procura para uma simulação de consulta apropriada.
- current – esse atributo está disponível apenas quando o atributo $method for modEntry. Seu valor é um objeto de entrada que representa a entrada

encontrada no sistema de destino, antes de ocorrer a modificação real. Essa é a entrada que o método findEntry, executado antes do método modEntry, retorna.

Uma Entrada de Trabalho Inicial (IWE) é transmitida para o AL do proxy quando ele for chamado. Quando o $method for findEntry, selectEntry ou getNextEntry, a IWE será uma cópia da entrada de trabalho a partir da chamada do AL. Em qualquer outro caso, a IWE será a entrada recuperada do procedimento OutputMap (também conhecida como entrada *conn*). Em particular, para a operação deleteEntry, a IWE é a entrada recuperada da operação findEntry anterior.

Depois que a execução do AL do proxy for feita e o $method for findEntry, a entrada do resultado será marcada para o atributo *conn*. Se ela estiver disponível, então supõe-se que esse atributo contenha todas as entradas encontradas na operação findEntry e de acordo com seu valor, os ganchos apropriados serão chamados, ou seja, no_match and multiple_match. Se nenhum atributo com o nome conn for encontrado, então a entrada de resultado da execução do AL do proxy será tratada como a entrada encontrada pelo findEntry $method. A entrada recuperada do AL do proxy que substitui o selectEntries $method é automaticamente mesclado à entrada de trabalho do AL de chamada. A entrada recuperada do AL do proxy que simula esses métodos que esperam uma entrada (ou seja, findEntry, getNextEntry, queryReply e executar) é enviada ao InputMap definido. Para todos os outros métodos que não se espera que retornem um resultado, a entrada do AL do proxy é ignorada.

Fluxo de Trabalho do Script de Simulação

Cada componente cujo estado de simulação está configurado como Com Script possui um Simulation Script (SS) definido na guia Simular.

Esta é a lista de objetos expostos a você para uso direto a partir de um SS:

- *work* – a entrada de trabalho.
- *conn* – a entrada recuperada da operação de consulta ou diretamente do OutputMap ou nula.
- *resEntry* – a entrada usada como um resultado se a operação que estiver sendo simulada exigir o retorno de um resultado; caso contrário, se o resultado não for usado, ela será nula.
- *current* – a entrada localizada no sistema de destino que será modificada ou nula.
- *search* – o objeto SearchCriteria definido pelo usuário ou nulo.
- *method* – um objeto String contendo o nome do método que está sendo substituído pelo SS.

A tabela a seguir explica os métodos que estão sendo simulados e mostra quando o SS será chamado.

Tabela 11. Solicitação de Método de Acordo com o Modo

Modo	Método
Conector: AddOnly	putEntry
Conector: Atualizar	modEntry, ou putEntry se a entrada não pôde ser localizada
Conector: Excluir	deleteEntry
Conector: Delta	modEntry, putEntry ou deleteEntry (depende da lógica interna)
Conector: Agente Iterativo	getNextEntry
Conector: CallReply	queryReply
Conector: Consultar	findEntry
Componente de Função	executar

Se um Conector estivesse no modo Servidor com um estado de simulação Com Script, ainda assim ele necessitaria receber um pedido de um cliente. O SS será chamado quando a resposta estiver para ser enviada.

A operação de consulta interna executada para os Conectores no modo Atualizar, Excluir e Delta será feita internamente e não permitirá a substituição de um SS como faz para o AL de proxy.

Os conectores nesse estado de simulação não executarão o gancho de substituição da operação, se houver algum.

Capítulo 5. Easy ETL

A perspectiva EasyETL no Editor de Configuração é uma maneira altamente especializada de examinar suas configurações do IBM Security Directory Integrator, dedicada a deixá-lo ativo e em execução rapidamente com projetos que giram em torno de tarefas simples para Extrair, Transformar e Carregar (ETL) dados em algum formato de banco de dados.

A perspectiva EasyETL mostra projetos EasyETL e o permite executar, abrir e criar novos projetos ETL. A perspectiva ETL pode ser mostrada escolhendo **Janela** > **Abrir Perspectiva** e, em seguida, selecionando a perspectiva Easy toETL.

Para iniciar o CE com essa perspectiva, inclua `-perspective com.ibm.tdi.rcp.perspective.etl` na linha de comandos do CE (`ibmditk`).

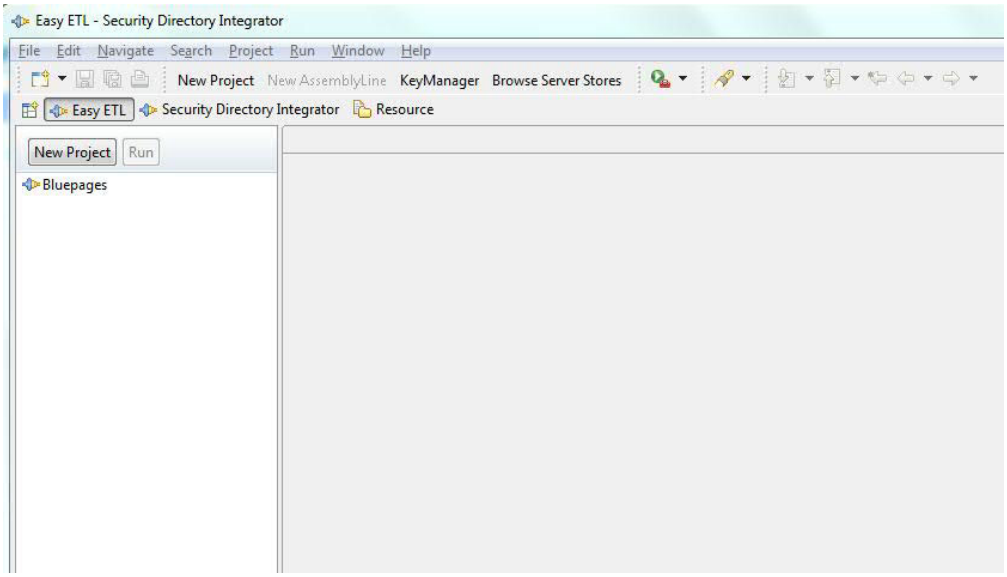


Figura 109. Janela Principal da EasyETL

Utilize o botão **Novo Projeto** para criar um novo projeto EasyETL. Um projeto EasyETL é um projeto normal do IBM Security Directory Integrator com um único AssemblyLine com dois conectores. Dê um clique duplo ou selecione o projeto e pressione a tecla Enter para abrir o editor ETL.

O menu de contexto para um projeto ETL tem os seguintes itens:

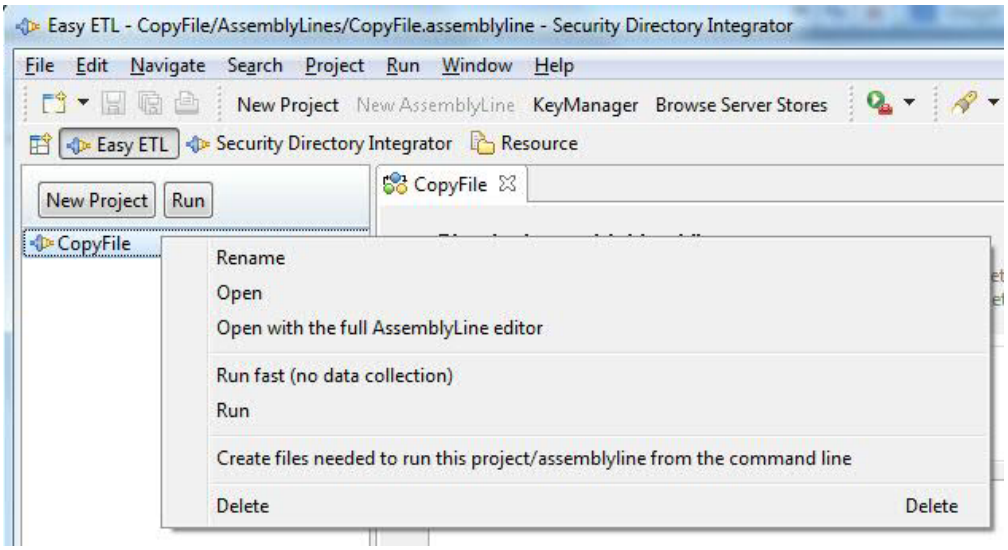


Figura 110. Menu de Contexto do Projeto EasyETL

- **Abrir** – Abra o projeto no editor
- **Abrir com Editor de AssemblyLine Integral** – Abra o projeto no editor de AssemblyLine avançado
- **Executar Rápido** – Execute o projeto sem coletar dados do AssemblyLine
- **Executar** – Execute o projeto e exiba dados coletados na visualização Coletor de Dados
- **Criar Arquivos...** – Gere arquivos necessários para executar o projeto a partir da linha de comandos
- **Renomear** – Renomeie o projeto
- **Excluir** – Exclua o projeto

O editor EasyETL mostra os dois conectores com uma tabela que mostra o mapeamento entre os dois conectores. A tela inicial mostra uma seleção vazia para ambos os conectores, conforme mostrado nesta figura:

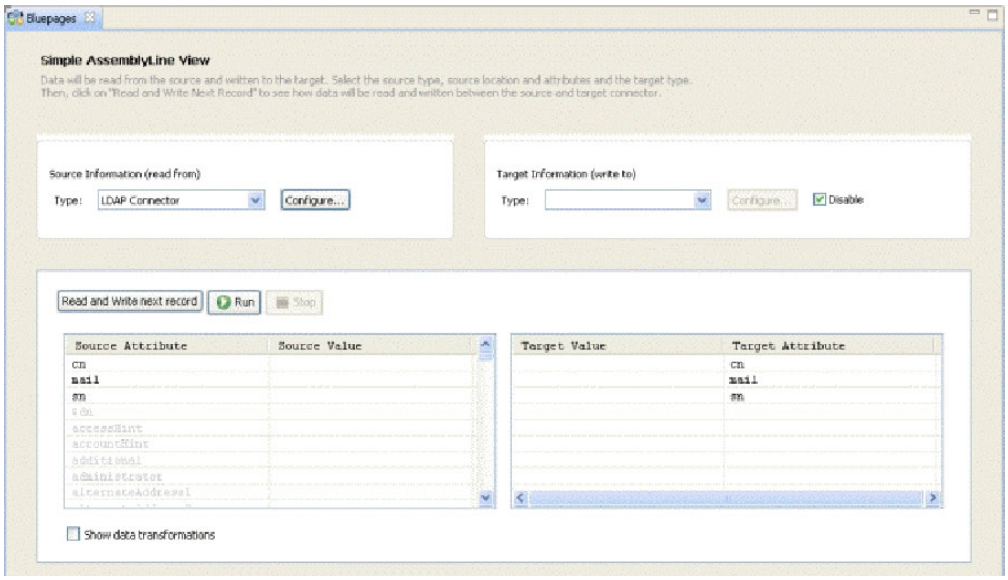


Figura 111. Janela do Projeto EasyETL Inicial

Normalmente você começa escolhendo o conector de origem. Existem quatro opções no tipo suspenso:

- Conector do Sistema de Arquivos
- Conector LDAP
- Conector do Banco de Dados (JDBC)
- Selecionar Conector...

Os três de cima são seleções rápidas, já que são comumente utilizados. A última opção, **Selecionar Conector...** apresenta o diálogo de seleção de conector padrão, onde você pode escolher qualquer conector que quiser. Entretanto, a lista de conectores é limitada aqueles que implementam o modo para o conector de origem (Agente Iterativo) e de destino (AddOnly).

Após o conector ser escolhido, você pode configurá-lo. O diálogo de configuração contém as formas para se configurar o conector e o analisador, caso seja necessário. Além disso, a tela *Delta* fica disponível para o conector de origem.

Se selecionar o Conector LDAP, você verá a seguinte janela:

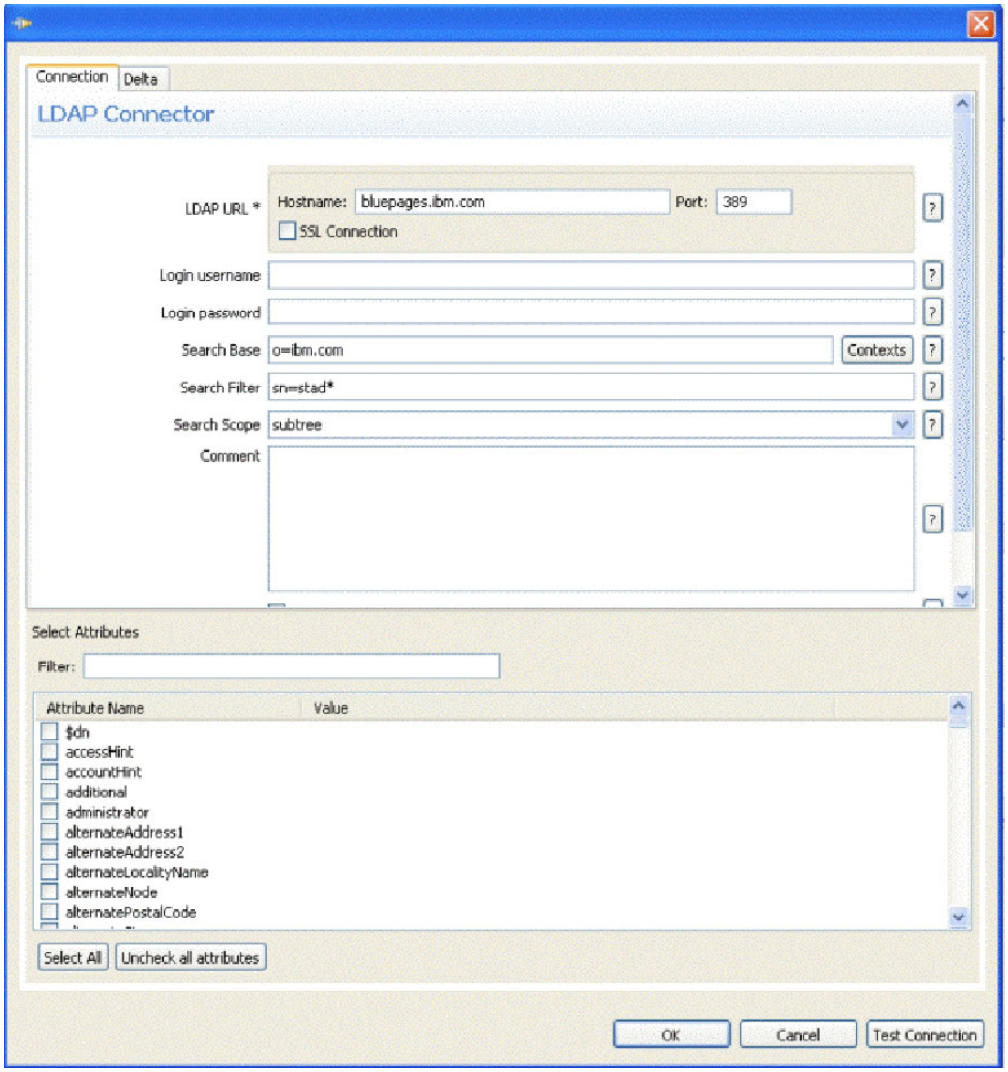


Figura 112. Conector LDAP em Easy ETL

Após descobrir os atributos no conector, você pode marcar os atributos que deseja ler no conector. Para o conector de destino, apenas a lista de itens de esquema está presente, já que o mapeamento é baseado nos atributos do conector de entrada.

Após o esquema e os atributos serem disponibilizados, eles serão mostrados na coluna do atributo de origem.

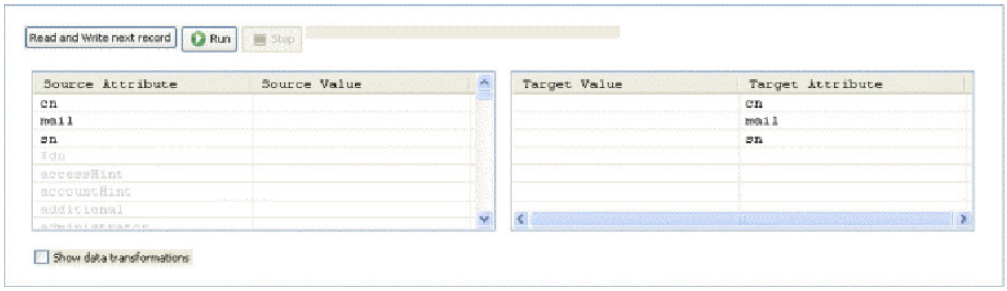


Figura 113. Mapeamento de Entrada/Saída

Itens esmaecidos são atributos que não são mapeados. Clique com o botão direito do mouse e escolha **Mapear Atributo** para mapear o atributo. Também é possível dar um clique duplo ou pressionar a tecla Enter para mapear a seleção atual. Na coluna do atributo de destino, você pode clicar e escolher um nome de atributo de saída diferente. Inversamente, é possível fazer o mesmo em um atributo mapeado para retorná-lo para a lista de atributos não mapeados.

Para customizar o mapeamento entre os dois atributos, marque a caixa de opção **Mostrar Transformações**. Isso incluirá uma nova tabela entre as tabelas de origem e de destino.

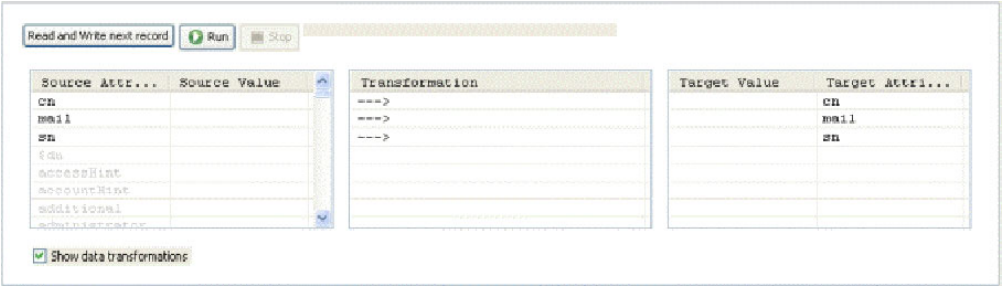


Figura 114. Mapeamento de Entrada/Saída com Transformações

A tabela de transformação mostrada tem setas que denotam cópias literalmente entre dois atributos. Clique duas vezes em um item de transformação para ativar o editor de JavaScript para esse mapa.

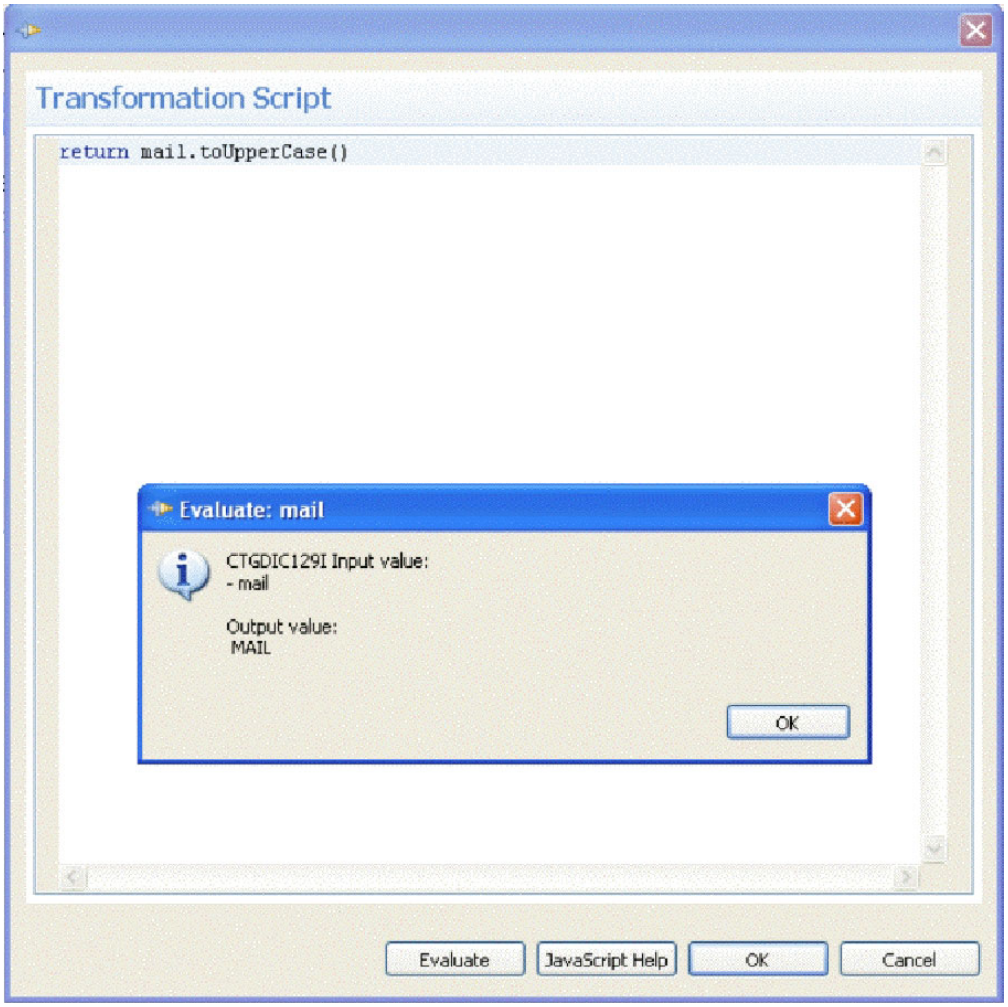


Figura 115. Script de Transformação

Insira o script que executará a transformação customizada do valor. Observe que todos os atributos mapeados no conector de origem estão disponíveis como beans de nível superior. Isso significa que você pode se referir diretamente ao `cn` em vez de utilizar a notação `work.cn`. O editor também está ciente da classe Java com base no que foi lido pela ação *Ler e Gravar Próximo*. A última entrada lida também é utilizada quando você testa o script com o botão **Avaliar**, portanto, a avaliação do script pode ser testada com relação aos dados ativos reais conforme mostrado na figura acima. A mensagem mostra o valor de entrada e o resultado do script de transformação (saída). O botão **Ajuda de JavaScript** é uma maneira rápida de acessar a página da ajuda para JavaScript.

O conector de destino tem uma caixa de opção rotulada **Desativar**. Essa caixa de opção, quando marcada, desativará o conector de saída e executará dump da entrada (após transformações customizadas) para o log do console.

Após configurar os conectores, você pode executar o AssemblyLine até a conclusão ou percorrendo um registro por vez através do AssemblyLine. Quando você percorre o AssemblyLine, a tabela reflete a última entrada lida e gravada no conector de destino. Quando você clica no botão **Executar**, o AssemblyLine é executado continuamente até a conclusão ou até você pressionar o botão **Parar**. Quando o botão parar é pressionado durante a execução, o AssemblyLine é interrompido e retorna o controle a você. Quando o botão parar é pressionado

enquanto você está no controle, o AssemblyLine é finalizado. Quando o AssemblyLine é finalizado, ele mostra um diálogo de conclusão com algumas estatísticas sobre a execução:

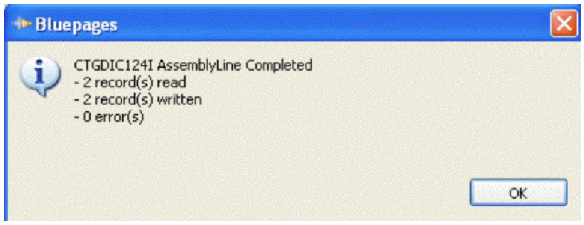


Figura 116. Diálogo de Conclusão

Capítulo 6. Armazenamento do Sistema

O Armazenamento do Sistema aborda as várias necessidades do IBM Security Directory Integrator para armazenamento persistente e, por padrão, usa o Apache Derby RDBMS (anteriormente conhecido como IBM Cloudscape) como sua tecnologia de armazenamento subjacente.

Outros bancos de dados relacionais, como o IBM DB2, podem ser usados para conter o Armazenamento do Sistema. O Armazenamento do Sistema poderá ser compartilhado por múltiplas instâncias de servidores IBM Security Directory Integrator se o banco de dados Derby for executado no modo em rede ou se um sistema de banco de dados relacional multiusuário for usado. Se o Derby for executado integrado a um servidor IBM Security Directory Integrator, ele não poderá ser compartilhado simultaneamente com outros servidores.

O Armazenamento do Sistema implementa três tipos de armazenamentos persistentes para componentes do IBM Security Directory Integrator:

- "Armazenamento de Propriedade de Usuário" na página 204
- "Armazenamento Delta" na página 204
- Tabelas de Sandbox

Cada armazenamento oferece seu próprio conjunto de recursos e comportamento interno, assim como uma interface que pode ser chamada, que os usuários podem acessar a partir dos seus scripts, por exemplo, para persistir suas próprias informações sobre dados e estado.

Você pode configurar as "Configurações de Armazenamento do Sistema" na página 175 para um projeto selecionando o projeto no Navegador do IBM Security Directory Integrator e selecionar **Configurações e Criação de Log de Soluções**. Em seguida, selecione a guia **Armazenamento do Sistema**.

Você pode configurar um Conector JDBC para acessar diretamente qualquer uma das tabelas no Armazenamento do Sistema, embora a mudança de dados nessas tabelas deva ser evitada, pois isso pode causar mau funcionamento de sua solução.

Atenção: Se você estiver executando o Derby integrado ao IBM Security Directory Integrator em vez de executá-lo no modo em rede como um servidor, certifique-se de **Fechar** o banco de dados novamente antes de tentar testar ou executar sua Configuração. Como o Config Editor inicializa uma instância separada do servidor, em execução em sua própria JVM, o Armazenamento do Sistema não está disponível nesse servidor. Ao fechar a janela Detalhes sobre o Armazenamento do Sistema, sua conexão com o banco de dados também é fechada.

Nota: Embora o recurso Caixa de Areia também use a tecnologia de Armazenamento do Sistema, você especifica um novo diretório de banco de dados para cada AssemblyLine.

Armazenamento de Propriedade de Usuário

O Armazenamento de Propriedades de Usuário é uma tabela de Armazenamento do sistema usada para manter objetos Java serializados associados a um valor de chave. Este é o local onde os parâmetros e propriedades do componente persistente, como o **Armazenamento de Estados do Agente Iterativo**, são mantidos, além dos dados que você armazena.

Por exemplo, ao configurar o parâmetro **Armazenamento de Estados do Agente Iterativo** para o Conector de Detecção de Mudanças do Active Directory, você está especificando o valor da chave que o Conector usa para salvar e restaurar o estado do Agente Iterativo. Se você desejar que a Iteração inicie com a primeira (ou última) entrada de mudança, simplesmente exclua a entrada de Armazenamento de Estados do Agente Iterativo no Armazenamento de Propriedades de Usuário; ou seja, clique em **Excluir** próximo ao parâmetro.

Você pode persistir seus próprios objetos com as seguintes chamadas de `sistema`:

system.setPersistentObject(*keyValue***,obj)**
: Salva o objeto **obj** no Armazenamento de Propriedades de Usuário usando o *keyValue* especificado. O objeto é retornado se tiver sido salvo com êxito, caso contrário, a função retorna **null**.

system.getPersistentObject(*keyValue***)**
: Retorna o objeto com o *keyValue* específico a partir do Armazenamento da Propriedade do Usuário. Se o *keyValue* não for localizado, a função retornará NULL.

system.deletePersistentObject(*keyValue***)**
: Exclui o objeto com o *keyValue* especificado no Armazenamento da Propriedade do Usuário. Essa função retorna o objeto que foi excluído, ou NULL se o *keyValue* não foi localizado.

Estes métodos acessam o Armazenamento da Propriedade do Usuário padrão.

Entretanto, é possível criar e usar seus próprios armazenamentos usando a *Fábrica de Armazenamento*.

Se você exibir o Armazenamento da Propriedade do Usuário a partir da janela Armazenamento do Sistema, observe que ele tem a seguinte definição de tabela:

Chave A chave exclusiva (512 caracteres)

Entrada
: O objeto associado à chave

Nota: Qualquer objeto a ser persistido no Armazenamento de Propriedades de Usuário deve ser serializável.

Armazenamento Delta

O Armazenamento Delta é encontrado na pasta **Tabelas Delta** do navegador Armazenamento do Sistema. Cada tabela representa uma definição de parâmetro de **Armazenamento Delta** (na guia **Delta** de um Repetidor). Há várias classes e métodos para trabalhar diretamente com o Armazenamento Delta, embora isso não seja recomendado. Para obter informações adicionais sobre o recurso Delta, consulte a seção Capítulo 7, “Deltas”, na página 209.

Métodos da Fábrica de Armazenamento

Os seguintes exemplos são métodos que podem ser usados com o Depósito de Informações do Armazenamento:

public static PropertyStore getDefaultPropertyStore () throws Exception;
Retorna o Armazenamento de Propriedades padrão.

public static PropertyStore getPropertyStore (String table) throws Exception;
Retorna o Armazenamento de Propriedade identificado por nome. Somente uma instância de um nome informado está presente por vez.

@param name
O nome do Armazenamento de Propriedades.

@return
O objeto de Armazenamento de Propriedades associado ao nome.

public static String getSystemDatabaseURL ();
Retorna o JDBC URL do Armazenamento do Sistema.

public static Connection getConnection () throws Exception;
Retorna um objeto de conexão para o banco de dados padrão.

public static Connection getConnection (String database) throws Exception;
Retorna um objeto de conexão para o banco de dados nomeado com o AutoCommit definido para TRUE.

@param database
O nome do banco de dados.

public static Connection getConnection (String database, boolean autoCommit) throws Exception;
Retorna um objeto de conexão para o banco de dados nomeado.

@param database
O nome do banco de dados.

@param autocommit
O sinalizador AutoCommit.

@return
Um objeto de conexão para o banco de dados nomeado.

public static boolean dropTable (Connection connection, String table);
Coloca uma tabela no banco de dados associado à conexão.

@param connection
O objeto de conexão obtido por `getConnection()`.

@param table
A tabela a ser eliminada.

public static void verifyTable (Connection connection, String table, Vector sql) throws Exception;
Verifica se uma tabela está acessível no banco de dados.

@param connection
O objeto de conexão obtido por `getConnection()`. Se nulo, uma conexão com a tabela padrão será obtida.

@param table
O nome da tabela a ser verificado.

@param sql
Um vetor de instruções SQL para criar a tabela se ela não existir.

public static Exception dropTable (String tableName);
Coloca uma tabela no banco de dados padrão.

@param tableName
O nome da tabela a ser eliminada.

public static byte[] serializeObject (Object obj) throws Exception;
Serializa um objeto para uma matriz de byte.

@param obj
O objeto a ser serializado.

@return
A matriz de byte que contém o objeto serializado.

public static Object deserializeObject (byte[] array) throws Exception;
Retira a serialização de uma matriz de byte em um objeto Java.

@param array
A matriz de byte com o objeto Java serializado.

@return
O objeto Java restaurado.

Métodos de Armazenamento de Propriedades

Os seguintes exemplos são métodos que podem ser usados com o Armazenamento de Propriedades:

public Object setProperty (String key, Object obj) throws Exception;
Inclui ou atualiza um valor no Armazenamento de Propriedades. Se uma atualização for executada, o valor antigo é retornado.

chave @param
O identificador exclusivo.

@param obj
O valor.

@return
O valor antigo no caso de uma atualização.

public Object getProperty (String key) throws Exception;
Retorna um valor no Armazenamento de Propriedades.

chave @param
O identificador exclusivo.

@return
O valor no armazenamento ou NULL se não localizado.

public Object removeProperty (String key) throws Exception;
Remove um valor no Armazenamento de Propriedades.

chave @param
O identificador exclusivo a ser removido.

@return
O valor antigo ou **null** se a chave não estiver na tabela.

Métodos UserFunctions (Objeto do Sistema)

A classe UserFunctions (por exemplo, o objeto de sistema) possui métodos adicionais definidos para obter ou configurar objetos no Armazenamento de Propriedades de Sistema:

public Object getPersistentObject (String key) throws Exception;
Esse método recupera um objeto nomeado a partir do Armazenamento de Propriedades de sistema padrão.

chave @param
A chave exclusiva.

public Object setPersistentObject (String key, Object value) throws Exception;
Esse método armazena um objeto nomeado no Armazenamento de Propriedades de sistema padrão.

chave @param
A chave exclusiva.

@param value
O objeto a ser armazenado (deve ser serializável Java).

@return
O objeto antigo, se houver algum.

public Object removePersistentObject (String key) throws Exception;
Esse método remove um objeto nomeado no Armazenamento de Propriedades de Sistema padrão.

chave @param
A chave exclusiva.

@return
O objeto antigo, se houver algum.

Capítulo 7. Deltas

O recurso Mecanismo Delta está disponível para Conectores no modo Agente Iterativo. Se ativado na guia Delta do Agente Iterativo, o recurso Mecanismo Delta utilizará o Armazenamento do Sistema para fazer uma captura instantânea dos dados sendo iterados. Cada Entrada lida com sucesso é comparada com o banco de dados de captura instantânea chamado Armazenamento Delta para ver o que foi alterado. Com base nas diferenças entre a Entrada lida e Entrada armazenada no Armazenamento Delta, uma nova Entrada chamada Entrada Delta será criada pelo Mecanismo Delta. Essa Entrada é identificada com códigos de operação delta especiais para indicar o que foi alterado e como.

O modo Delta é um modo do Conector que permite que um conector "entenda" e utilize os códigos de operação delta. Um conector neste modo utiliza os códigos de operação delta de uma Entrada Delta recebida para determinar qual tipo de mudança precisa ser aplicado no sistema conectado. O modo Delta suporta todos os tipos de modificações - incluir, modificar e excluir. Esse modo é utilizado para facilitar a sincronização entre diferentes sistemas (por exemplo, sincronizar dois servidores LDAP em máquinas diferentes).

Atenção: O Mecanismo Delta apresenta um repositório local subjacente para armazenamento de capturas instantâneas de dados para computar alterações durante o processo de sincronização. A origem de dados que está sendo varrida para procurar alterações torna-se a mestra em um relacionamento de mestre-escravo e é, por isso, vital que todas as alterações feitas no escravo (por exemplo, o armazenamento Delta) sejam feitas utilizando o mecanismo Delta, em vez de manipular a tabela de banco de dados subjacente diretamente. Caso contrário, as informações de captura instantânea Delta que o IBM Security Directory Integrator mantém tornam-se inconsistentes e o Mecanismo Delta falha.

Recursos Delta

Os recursos Delta no IBM Security Directory Integrator são os seguintes:

Entrada Delta

Esse é um objeto Entrada regular que foi identificado com códigos de operação delta especiais. Esses códigos descrevem o tipo de mudança (*add*, *modify*, *delete* ou *unchanged*) e podem ser designados em níveis diferentes - nível Entrada, Atributo ou Valor do Atributo.

Componentes Produzindo Entradas Delta

- Mecanismo Delta - detecta mudanças em uma origem de dados. É útil quando a origem de dados em si não fornece acesso conveniente às mudanças (por exemplo, mecanismo de notificação de mudança ou log de mudanças). Mudanças são detectadas através da comparação do estado atual da origem de dados com uma captura instantânea histórica. A captura instantânea é salva em um repositório chamado "Armazenamento Delta". Fisicamente, o Armazenamento Delta consiste em inúmeras Tabelas Delta localizadas no Armazenamento do Sistema.

  Na próxima vez que os dados forem lidos, eles serão comparados com aqueles já salvos no Armazenamento Delta. Após as informações da mudança serem

computadas, uma Entrada Delta será criada e retornada pelo Conector. Essa Entrada Delta pode ser utilizada para transferir as mudanças detectadas para outros sistemas conectados através do uso de Conectores no modo *Atualizar*, *AddOnly*, *Excluir* ou *Delta*.

- Conectores de Detecção de Mudanças – esses são os Conectores LDAP, o Conector de Detecção de Mudanças RDBMS e o Conector de Detecção de Mudanças Domino.
- Analisador LDIF e DSMLv2 - durante a leitura ou gravação, o Analisador LDIF e DSMLv2 suporta identificação Delta no nível Entrada, Atributo e Valor do Atributo.

Componentes Consumindo Entradas Delta

- Modo Delta - esse modo do Conector facilita soluções de sincronização entre sistemas. Ele pode ser utilizado para aplicar todos os tipos de mudanças em um sistema conectado. O modo Delta é o único modo que requer (e utiliza) Entradas Delta. Esse modo detecta tipos de mudanças utilizando códigos de operação delta de uma Entrada.
- Atualizar Conectores com Alterações de Cálculo - Conectores no modo Atualizar com o parâmetro “Alterações de Cálculo” ativado não acionam a lógica “Alterações de Cálculo” se a Entrada recebida for identificada como delta.

Entrada Delta

Uma Entrada Delta é uma Entrada que possui todos os recursos e funções de uma Entrada regular. Além disso, a Entrada Delta contém os códigos de operação delta. Eles indicam o tipo de mudança aplicada na Entrada – incluir, excluir, modificar ou inalterada. Os códigos de operação delta podem ser anexados às Entradas, aos Atributos e aos valores para refletir suas mudanças específicas.

Visão Geral

O processo de designação de códigos de operação delta é chamado identificação de delta, e os códigos delta são referidos como códigos de operação e tags delta. Em poucas palavras, a Entrada Delta é, de fato, uma Entrada regular identificada como Delta. Veja aqui um exemplo de uma Entrada regular em uma Entrada Delta.

Entrada Regular:

```
{
 "#type": "generic",
 "#count": 3,
 "UserName":
  "#type": "replace",
  "#count": 1,"tanders",
 "FullName":
  "#type": "replace",
  "#count": 1,"Teddy Anderson",
 "id":
  "#type": "replace",
  "#count": 1,"66"
}
```

Entrada Delta:

```
{
 "#type": "modify",
 "#count": 3,
 "@delta.old": "{
 "UserName": "manders",
 "FullName": "Mary Anderson",
```

```
 "id": "66"
 }",
 "UserName": [
  "#type": "modify",
  "#count": 2,
  "tanders",
  "manders"
 ],
 "FullName": [
  "#type": "replace",
  "#count": 1,
  "Mary Anderson"
 ],
 "id":
  "#type": "unchanged",
  "#count": 1,"66"
}
```

O processo de avaliação de códigos delta vai de cima a baixo ou do nível Entrada → nível Atributo → nível Valor do Atributo. A operação no nível mais alto tem precedência.

Se uma operação de Entrada for *excluir*, todas as outras tags delta serão ignoradas. Se for *substituir*, *modificar* ou *incluir*, a avaliação continuará com as tags delta de Atributos.

Se um Atributo for identificado como *delete*, *add* ou *replace*, as tags delta desses valores serão ignoradas. Somente se um Atributo for identificado como *modify*, as tags delta de Valores do Atributo serão consideradas. Essas tags delta indicam os Valores do Atributo que têm códigos de operação diferentes (por exemplo, alguns deles são incluídos e outros excluídos).

As tags delta do Valor do Atributo têm o seguinte significado no contexto da identificação Delta:

- *add* – o valor é incluído na lista de valores para o Atributo especificado;
- *delete* – o valor é removido da lista de valores do Atributo especificado;
- *replace* – o valor é substituído; essa é a tag delta padrão.

Entradas Delta são produzidas pelos seguintes componentes:

1. Conectores no modo Agente Iterativo com Delta ativado;
2. Conectores de Detecção de Alterações;
3. Analisador LDIF e DSMLv2 durante a leitura.

Eles são consumidos por esses componentes:

1. Conectores no modo Delta;
2. Analisador LDIF e DSMLv2 durante a gravação;
3. Conectores no modo Atualizar.

Obtendo e Configurando Códigos de Operação Delta Utilizando Script

A identificação Delta é suportada nos níveis Entrada, Atributo e Valor do Atributo. Aqui você verá como obter/configurar a operação Entrada utilizando script:

```
var entryOper = work.getOperation();  //obter operações de Entrada como cadeia (por exemplo, 'add')
var entryOp = work.getOp();      //obter operações de Entrada como caractere (por exemplo, 'a')

work.setOperation("modify");      //configurar operação de Entrada
work.setOp ('m');
```

Se quiser configurar/obter os sinalizadores Delta para um Atributo, é possível fazer isso com o seguinte código:

```
var attr = work.getAttribute("sn");   // obter objeto Atributo

var attrOper = attr.getOperation();   // obter operação delta como cadeia (por exemplo, 'replace')
var attrOp = attr.getOper();      // obter operação delta como caractere (por exemplo, 'r')

attr.setOperation("replace");     // configurar operação delta do Atributo
attr.setOper('r');
```

Tags Delta no nível Valor do Atributo podem ser configuradas/obtidas através deste script:

```
var attr = work.getAttribute("sn");   // obter objeto Atributo

var valOper = attr.getValueOperation(0);// obter operação delta de valor para primeiro valor
var valOp = attr.getValueOper(0);

attr.setValueOperation(1, "add");    // configurar operação delta de valor para segundo valor
attr.setValueOper(1, 'a');
```

Produzindo Entradas Delta

Entradas Delta podem ser geradas através do recurso Delta ou de um analisador adequado de Conectores no modo Agente Iterativo ou Conectores de Detecção de Alterações do IBM Security Directory Integrator.

Em particular, as Entradas identificadas como Delta são retornadas pelos seguintes componentes:

- Conector de Detecção de Alteração do Active Directory
- Conector de Detecção de Mudanças do Domino
- Conector de Log de Mudanças do IBM Security Directory Server
- Conector de Detecção de Alterações do RDBMS
- Conector de Detecção de Alteração do Sun Directory
- Conector do Log de Mudanças do z/OS LDAP

  Nota: O sistema operacional z/OS não é suportado a partir do IBM Security Directory Integrator Versão 7.2.
- Analisador DSMLv2
- Analisador LDIF

Recurso Delta para Modo Agente Iterativo

Conectores no modo Agente Iterativo podem gerar entradas Delta. Esse recurso utiliza o Mecanismo Delta e o Armazenamento Delta para detectar alterações.

Mecanismo Delta

O Mecanismo Delta permite ler por meio de uma origem de dados e detecta alterações feitas desde a última vez que você fez isso. Dessa forma, você pode detectar novas entradas, entradas alteradas e até mesmo entradas excluídas. Para certas origens de dados (como arquivos LDIF e servidores LDAP), o IBM Security Directory Integrator pode até detectar se atributos e valores dentro das entradas foram alterados. É possível configurar as definições Delta em Conectores somente no modo Repetidor.

O Mecanismo Delta sabe se Entradas ou Atributos foram incluídos, alterados ou excluídos, mantendo uma cópia local de cada Entrada em um armazenamento persistente que faz parte do Armazenamento do Sistema. Esse repositório local é chamado *Armazenamento Delta* e consiste em *Tabelas Delta*. Cada vez que o AssemblyLine é executado, o Mecanismo Delta compara os dados sendo iterados com essa cópia na Tabela Delta. Quando uma mudança é detectada, o Conector retorna uma *Entrada Delta*.

Nota: Não modifique manualmente tabelas de Armazenamento Delta. Caso contrário, as informações de captura instantânea Delta ficarão inconsistentes e o Mecanismo Delta falhará.

Nota: Em versões anteriores à IBM Security Directory Integrator V6.1, as capturas instantâneas gravadas no Armazenamento Delta durante o processamento do mecanismo Delta foram confirmadas imediatamente. Como resultado, o Mecanismo Delta consideraria uma Entrada alterada como tratada, mesmo que o processamento da seção Fluxo do AL falhasse. Essa limitação é abordada através do parâmetro de confirmação na guia Delta do Conector. A configuração desse parâmetro controla quando o Mecanismo Delta confirma capturas instantâneas obtidas de dados recebidos para o Armazenamento do Sistema.

Nome de Atributo Exclusivo

Para o Mecanismo Delta poder identificar exclusivamente cada Entrada, você deverá especificar um Atributo exclusivo para utilizar como uma chave Delta. Os valores desse atributo devem ser exclusivos na origem de dados utilizada. Você pode especificar a chave Delta na guia Delta do Conector inserindo ou selecionando um nome de atributo no parâmetro Nome de Atributo Exclusivo. Esse atributo deve ser encontrado no Mapa de Entrada do Agente Iterativo e pode ser um atributo lido a partir do sistema conectado ou um atributo computado (utilizando Mapeamento de Atributo).

Também é possível especificar vários atributos separando com um sinal de mais (+):

```
LastName+FirstName+BirthDate
```

Pelo menos um dos atributos especificados no parâmetro Nome de Atributo Exclusivo deve conter um valor. Quando vários atributos são especificados, seus valores de cadeia são concatenados em uma cadeia, que, então, se torna o identificador Delta exclusivo. Atributos sem valores (por exemplo, em branco ou NULL) são ignorados quando a chave Delta é construída para uma Entrada.

Armazenamento Delta

O Armazenamento Delta está localizado fisicamente no Armazenamento do Sistema. Ele consiste em uma Systable Delta (DS) e em uma ou mais Tabelas Delta. Cada Tabela Delta é utilizada para o Armazenamento Delta de um Conector de Agente Iterativo diferente com Delta ativado.

Embora tabelas de Armazenamento Delta possam ser acessadas com o conector JDBC e o Conector de Armazenamento do Sistema, não é recomendável alterá-las sem um profundo entendimento de como essas tabelas são estruturadas e tratadas pelo Mecanismo Delta.

Estrutura da Tabela Delta

Cada Tabela Delta (DT) contém informações sobre cada Entrada processada pelo Mecanismo Delta para um determinado Conector. Uma Systable Delta (DS) mantém uma lista de todas as Tabelas Delta atualmente em uso pelo Armazenamento Delta.

- Systable Delta – A Systable Delta (DS) contém informações sobre cada Tabela Delta (DT) no Armazenamento do Sistema. A finalidade da DS é manter o contador de sequência para cada DT. A estrutura para a DS é a seguinte:

Tabela 12. Estrutura da Delta Systable

Coluna	Tipo	Descrição
ID	Varchar	O Identificador da DT (nome)
SequenceID	Int	O ID de sequência a partir da última execução
Versão	Int	A versão da DS (1)

- Tabela Delta - Cada Conector que solicita um armazenamento Delta precisa especificar um identificador Delta exclusivo para ser associado ao Conector. Este identificador é utilizado também como o nome da Tabela Delta no Armazenamento do Sistema. A estrutura da Tabela Delta é a seguinte:

Tabela 13. Estrutura da Tabela Delta

Coluna	Tipo	Descrição
ID	Varchar	O valor exclusivo identificando uma Entrada
SequenceID	Int	O número de sequência da Entrada
Entrada	Long Varbinary	O objeto de Entrada serializado

Processo Delta

Dada a estrutura de Armazenamento Delta acima, o número de sequência é utilizado para determinar quais entradas não fazem mais parte do conjunto de dados de origem. Cada vez que um AssemblyLine é executado, o número de sequência para a Tabela Delta utilizado em particular pelo Conector é lido da Systable Delta. Então ele é incrementado e esse valor incrementado será utilizado para marcar as entradas atualizadas durante a execução do AssemblyLine inteiro.

O processo do Mecanismo Delta funciona em dois passos.

1. Ler → Consultar → Comparar → Atualizar → Configurar Atual ID de Sequência
   a. O Agente Iterativo lê entradas da origem de dados de entrada.
   b. O processo Delta procura a Entrada correspondente na Tabela Delta utilizando o valor do atributo exclusivo.
   c. Se uma correspondência for localizada, o processo Delta comparará cada Atributo (e seus valores) para determinar se houve alguma modificação na Entrada. Com base no resultado da comparação, o Mecanismo Delta retorna a Entrada Delta identificada com códigos de operação relevantes: *modify* ou *unchanged*:
      - Entrada Modify - a Entrada que foi lida e a Entrada correspondente da Tabela Delta são consideradas diferentes; a Entrada é atualizada na Tabela Delta
      - Entrada Unchanged - a Entrada que foi lida e a Entrada correspondente da Tabela Delta são consideradas iguais.
   d. Se uma correspondência não foi localizada na Tabela Delta, a Entrada será tratada como nova:
      - Entrada Add - a Entrada é incluída na Tabela Delta.

e. Nos casos c. e d., o valor do número de sequência na tabela Delta é atualizado com o número de sequência utilizado para a execução do AssemblyLine atual.

2. Verifique os dados com (ID de Sequência < ID de Sequência Atual) → Marcar como Excluído

   Após o Fim dos Dados ser atingido pelo Agente Iterativo, o Mecanismo Delta fará uma segunda passagem pela Tabela Delta procurando aquelas entradas não acessadas durante a primeira passagem. Essas Entradas são facilmente reconhecidas porque seu número de sequência não é atualizado com o número de sequência atual. Portanto, quaisquer Entradas na Tabela Delta com um número de sequência inferior ao número de sequência atual serão consideradas entradas excluídas e serão retornadas como excluídas.

   Nota: Essa passagem acontece apenas quando a iteração através dos dados de entrada é concluída com sucesso. Se por algum motivo ocorrer um erro durante essa iteração, nenhuma Entrada será identificada como excluída e retornada pelo AssemblyLine ou removida da Tabela Delta. Isso não afetará a origem de dados original, e na próxima vez que o AssemblyLine for executado com sucesso, as Entradas excluídas serão processadas corretamente.

Bloqueio de Linha

Esse parâmetro está disponível na guia Delta para conectores do Agente Iterativo e configuração do Componente de Função Delta. Ele permite que você configure o nível de isolamento de transação utilizado pela conexão estabelecida com o banco de dados de Armazenamento Delta. A configuração de um nível de isolamento maior reduz anomalias da transação conhecidas como 'leituras sujas', 'leituras não repetíveis' e 'leituras fantasma' usando bloqueios de linha e de tabela. Esse parâmetro tem os seguintes valores:

READ_UNCOMMITTED
: Corresponde a `java.sql.Connection.TRANSACTION_READ_UNCOMMITTED`; indica que leituras sujas, leituras não repetíveis e leituras fantasma podem ocorrer. Esse nível permite que uma linha alterada por uma transação seja lida por outra transação antes que quaisquer mudanças nessa linha sejam confirmadas (uma "leitura suja"). Se alguma das mudanças for retrocedida, a segunda transação terá recuperado uma linha inválida.

READ_COMMITTED
: Corresponde a `java.sql.Connection.TRANSACTION_READ_UNCOMMITTED`; indica que leituras sujas são evitadas; leituras não repetíveis e leituras fantasma podem ocorrer. Esse nível só proíbe uma transação de ler uma linha contendo mudanças não confirmadas.

REPEATABLE_READ
: Corresponde a `java.sql.Connection.TRANSACTION_REPEATABLE_READ`; indica que leituras sujas e leituras não repetíveis são evitadas; leituras fantasma podem ocorrer. Esse nível proíbe uma transação de ler uma linha contendo mudanças não confirmadas, além de proibir a situação na qual uma transação lê uma linha, uma segunda transação altera a linha e a primeira transação relê a linha, obtendo valores diferentes na segunda vez (uma "leitura não repetível").

SERIALIZABLE
: Corresponde a `java.sql.Connection.TRANSACTION_SERIALIZABLE`; indica que leituras sujas, leituras não repetíveis e leituras fantasma são evitadas. Esse nível inclui as proibições em TRANSACTION_REPEATABLE_READ, além

de proibir a situação na qual uma transação lê todas as linhas que satisfazem uma condição WHERE, uma segunda transação insere uma linha que satisfaz essa condição WHERE e a primeira transação relê a mesma condição, recuperando a linha "fantasma" adicional na segunda leitura. Essa geralmente é a opção mais lenta, mas a mais segura, e é o valor-padrão para o parâmetro **Bloqueio de Linha**.

Para obter mais informações sobre níveis de isolamento de transação, consulte a documentação online da interface java.sql.Connection: http://docs.oracle.com/javase/1.6.0/docs/api/java/sql/Connection.html.

Cada servidor de banco de dados configura um nível de isolamento de transação padrão; o valor padrão para Apache Derby, Oracle e Microsoft SQL Server é TRANSACTION_READ_COMMITTED. Entretanto, o valor-padrão do parâmetro **Bloqueio de Linha** de SERIALIZABLE irá substituí-lo ao utilizar um componente Delta (ou seja, a funcionalidade Delta nos Conectores do Agente Iterativo ou o Componente de Função Delta).

Alguns servidores de banco de dados podem não suportar todos os níveis de isolamento de transação, portanto, consulte a documentação do banco de dados específico para obter informações exatas sobre os níveis de isolamento de transação suportados.

Nota: Os níveis de isolamento de transação são mantidos pelo servidor de banco de dados em si para cada conexão estabelecida com o banco de dados. Portanto, quando um componente Delta (com **Nível de Isolamento de Transação** configurado como REPEATABLE_READ ou SERIALIZABLE e parâmetro **Confirmar** configurado como `No Fechamento do Conector`) iniciar sua transação, todas as outras consultas tentando modificar os mesmos dados serão bloqueadas. Isso significa que outros componentes que precisam modificar os mesmos dados terão que esperar até o primeiro componente confirmar sua transação no término. Essa espera pode fazer as consultas SQL emitidas expirarem e deixarem os dados inalterados.

Além disso, quando um componente tem o parâmetro **Confirmar** configurado como `Sem Confirmação Automática`, normalmente você deve confirmar as transações de uma maneira na qual os outros componentes não vão esperar muito tempo para fazer uma modificação.

Detectar ou Ignorar Mudanças Apenas em Atributos Específicos

Os parâmetros **Lista de Atributos** e **Modo de Detecção de Alterações** configuram a capacidade do Mecanismo Delta de detectar mudanças apenas em atributos específicos, e não em todos os atributos recebidos.

O parâmetro **Lista de Atributos** é uma lista de atributos separados por vírgula que serão afetados pelo **Modo de Detecção de Alterações**. Esse parâmetro **Modo de Detecção de Alterações** especifica como as mudanças nesses atributos serão tratadas. Ele tem três valores:

IGNORE_ATTRIBUTES
("Ignorar Mudanças para os Seguintes Atributos") – Mudanças em cada atributo especificado no parâmetro **Lista de Atributos** serão ignoradas durante o processo de mudança de cálculo.

DETECT_ATTRIBUTES
("Detectar Mudanças para os Seguintes Atributos") – Esta opção tem o efeito oposto – as únicas mudanças detectadas serão nos atributos listados no parâmetro **Lista de Atributos**.

DETECT_ALL
("Utilizar todos os Atributos para Detecção de Alterações") – Instrui o Mecanismo Delta a detectar mudanças em todos os atributos. Quando essa opção é selecionada, o parâmetro **Lista de Atributos** é desativado, já que nenhuma lista de atributos afetados é necessária.

Caso de Uso de Exemplo

Durante o uso do Mecanismo Delta, às vezes as entradas recebidas contêm atributos que você não considera importantes e quer ignorar. Nesses casos, esses atributos não devem afetar o resultado do cálculo Delta, pois quando várias Entradas são diferenciadas apenas por esses atributos, isso leva a atualizações desnecessárias da tabela Armazenamento Delta.

A solução para esse caso é utilizar os parâmetros **Lista de Atributos** e **Modo de Detecção de Alterações**

Veja aqui um exemplo de cenário no qual dois AssemblyLines estão recebendo entradas do log de mudanças de duas réplicas de um servidor LDAP e essas mudanças são aplicadas no Armazenamento Delta. Para ilustrar isso, vamos utilizar as seguintes entradas do log de mudanças de exemplo:

Entry1:

```
Entry attributes:
 targetdn (replace): 'cn=Niki,o=IBM,c=us'
 changetime (replace): '20071015094646'
 $dn (replace): 'changenumber=78955,cn=changelog'
 ibm-changeInitiatorsName (replace): 'CN=ROOT'
 changenumber (replace): '78955'
 objectclass (replace): 'top' 'changelogentry' 'ibm-changelog'
 changetype (replace): 'modify'
 cn (replace): 'Niki' 'Niky'
 changes (replace): 'replace: cn
       cn: Niki
       cn: Niky
       -
       '
```

Entry2:

```
Entry attributes:
 targetdn (replace): 'cn=Niki,o=IBM,c=us'
 changetime (replace): '20071015094817'
 $dn (replace): 'changenumber=10076,cn=changelog'
 ibm-changeInitiatorsName (replace): 'CN=ROOT'
 changenumber (replace): '10076'
 objectclass (replace): 'top' 'changelogentry' 'ibm-changelog'
 changetype (replace): 'modify'
 cn (replace): 'Niki' 'Nikolai'
 changes (replace): 'replace: cn
       cn: Niki
       cn: Nikolai
       -
      '
```

Entry3:

```
Entry attributes:
 targetdn (replace): 'cn=Niki,o=IBM,c=us'
 changetime (replace): '20071037454817'
 $dn (replace): 'changenumber=112,cn=changelog'
 ibm-changeInitiatorsName (replace): 'CN=ADMIN'
 changenumber (replace): '112'
 objectclass (replace): 'top' 'changelogentry' 'ibm-changelog'
 changetype (replace): 'modify'
 cn (replace): 'Niki' 'Nikolai'
 changes (replace): 'replace: cn
       cn: Niki
       cn: Nikolai
       -
          '
```

Os atributos modificados são marcados em **negrito** e os atributos que podem ser ignorados são marcados em *itálico*. Os atributos ignorados (como changenumber, changetime, entre outros) não serão considerados quando você comparar a Entrada recebida com a Entrada armazenada. Portanto, esses atributos devem ser listados no parâmetro **Lista de Atributos**. Para especificar que queremos ignorá-los, o parâmetro **Modo de Detecção de Alterações** precisa ser configurado como `Ignorar Mudanças para os Seguintes Atributos`.

Esse é o fluxo de trabalho quando o AssemblyLines recebe entradas:

1. Quando AL1 receber Entry1, ele será retornado como *modify* e salvo na tabela Armazenamento Delta.
2. Quando AL2 receber Entry2, seus atributos changetime, $dn, bm-changeInitiatorsName, changenumber serão modificados, mas serão ignorados. Entretanto, os atributos cn e changes também são modificados e, portanto, a Entrada Delta resultante será identificada como *modify* e salva na tabela Armazenamento Delta.
3. Quando AL2 receber Entry3, seus atributos changetime, $dn, bm-changeInitiatorsName, changenumber serão modificados, mas serão ignorados. O restante dos atributos será igual, portanto, a Entrada Delta resultante será identificada como *unchanged* e será retornada para o AssemblyLine (apenas se o parâmetro **Retornar Inalterado** for marcado) ou ignorada. A Entrada Delta retornada será idêntica à Entry3 recebida. Nesse caso, o Armazenamento Delta não será atualizado. Se os parâmetros Lista de Atributos e Modo de Detecção de Alterações não foram utilizados, a última Entry3 será identificada como *modify* e salva no Armazenamento Delta.

Conectores de Detecção de Alterações

Os Conectores de Detecção de Alterações alavancam informações no sistema conectado para detectar alterações e são utilizados no Modo Agente Iterativo ou Servidor, dependendo do Conector. Por exemplo, o modo Agente Iterativo é utilizado para vários Conectores de Detecção de Alterações, como aqueles para LDAP, RDBMS, Active Directory e Notes/Domino. Eles são designados para se comportarem de uma maneira comum, bem como para fornecer as mesmas etiquetas de parâmetro para configurações comuns.

Os Conectores de Detecção de Mudanças no IBM Security Directory Integrator são:

- Log de Mudanças do IBM Security Directory Server
- Detecção de Alterações do AD (Active Directory)
- Detecção de Alterações do Domino
- Detecção de Alterações do Sun Directory (openLDAP, SunOne, iPlanet, etc.)
- Detecção de Alterações do RDBMS (DB2, Oracle, SQL server, etc.)

- Log de Mudanças do z/OS LDAP

  Nota: O sistema operacional z/OS não é suportado a partir do IBM Security Directory Integrator Versão 7.2.

O recurso do mecanismo Delta relata alterações inteiramente específicas para os valores individuais de atributos. A identificação de Delta no nível Valor do Atributo também está disponível durante a análise com os Analisadores LDIF ou DSMLv2. O Analisador LDIF é usado internamente pelo Conector de Log de Mudanças do IBM Security Directory Server, Conector de Detecção de Mudanças do Sun Directory e Conector de Log de Mudanças LDAP do z/OS, portanto, esses conectores também suportam a identificação Delta no nível AttributeValue. O restante dos conectores de detecção de alterações limita-se simplesmente a relatar se uma Entrada inteira foi incluída, modificada ou excluída. Para obter mais informações sobre o suporte à identificação de Delta de um determinado componente, consulte a descrição específica desse componente no *Referência*.

Em alguns casos, AssemblyLines de longa execução podem precisar processar as mesmas entradas mais de uma vez. Essas entradas terão chaves delta duplicadas e farão o AssemblyLine emitir uma exceção. Para permitir chaves delta duplicadas, é possível selecionar a caixa de seleção **Permitir Chaves Delta Duplicadas** na guia Delta do Agente Iterativo. Isso significa que entradas duplicadas podem ser manipuladas pelo AssemblyLines que têm Conectores do Agente Iterativo com Delta ativado ou Conectores de Detecção de Alterações e Conectores no modo Delta.

Nota: É possível ter, por exemplo, um AssemblyLine com inúmeros Conectores no modo Delta e de Log de Mudanças. Nesse caso, se o Conector no modo Delta estiver apontando para o mesmo sistema subjacente que o Conector do Log de Mudanças, a operação Delta poderia ativar o Log de Mudanças novamente. Como não há uma maneira de diferenciar entre alterações recém-recebidas e aquelas ativadas pelo mecanismo Delta, você deverá considerar cuidadosamente o seu cenário a fim de não entrar em um loop sem fim.

As informações delta computadas pelo mecanismo Delta são armazenadas no objeto Entrada de Trabalho e, dependendo do componente ou do recurso de Detecção de Alterações utilizado, podem ser armazenadas como um código de operação *Nível de Entrada*, *Nível de Atributo* ou *Nível do Valor de Atributo*.

Consumindo Entradas Delta

Entradas Delta no AssemblyLine podem ser utilizadas ("consumidas") pelos Conectores no Modo Delta e Conectores no Modo Atualizar na lógia Computar Alterações.

O Modo Delta está disponível nos seguinte Conectores:

- Conector JDBC
- Conector DSMLv2 SOAP
- Conector JNDI
- Conector LDAP

Conectores de Modo Delta

O modo Delta destina-se a simplificar o aplicativo de alterações a dados, fornecendo modificações incrementais ao sistema conectado, baseado em códigos de operação delta. Modificações incrementais significam gravar apenas os valores específicos que foram alterados.

Em primeiro lugar, o modo Delta manipula todos os tipos de deltas: inclusões, modificações e exclusões. O modo Delta requer o recebimento de uma Entrada Delta para operar. Portanto, quando você utiliza um Conector no modo Delta, ele deve ser combinado com os componentes que produzem Entradas Delta; eles são Conectores do Agente Iterativo com Delta ativado, Conector ou Conectores de Detecção de Alterações utilizando um Analisador LDIF ou DSMLv2. Por exemplo, é possível sincronizar dois sistemas com o uso de apenas dois Conectores: um Conector de Detecção de Alterações na seção Alimentação para selecionar as mudanças e um segundo Conector no modo Delta para aplicar essas mudanças em um sistema de destino.

Além disso, o modo Delta aplicará as informações delta no nível mais inferior suportado pelo próprio sistema de destino. Isso é feito verificando primeiro a Interface de Conector para saber que nível de modificação incremental é suportado pela origem de dados. Se você estiver trabalhando com um diretório LDAP, o modo Delta realizará inclusões e exclusões do Valor do Atributo. No contexto de um RDBMS (JDBC) tradicional, excluir e depois incluir um valor de coluna não faz sentido, portanto, isso é tratado como uma substituição de valor para esse Atributo.

Além disso, modificações incrementais são tratadas automaticamente pelo modo Delta para aquelas origens de dados que suportam essa funcionalidade. Se a origem de dados oferecer chamadas otimizadas para manipular modificações incrementais e essas modificações forem suportadas pela Interface de Conector, o modo Delta usará as chamadas otimizadas. Por outro lado, se o sistema conectado não oferecer mecanismos de atualização delta "inteligentes", o modo Delta simulará esses mecanismos da melhor maneira possível, realizando consultas pré-atualização (como no modo Atualizar), cálculos de alterações e a aplicação subsequente das alterações detectadas.

Nota: Os únicos Conectores que suportam modificações incrementais são os Conectores LDAP, já que os diretórios LDAP fornecem essa funcionalidade.

Os recursos Delta no IBM Security Directory Integrator são projetados para facilitar as soluções de sincronização utilizando não apenas origens de dados que fornecem mecanismo de detecção, mas também, por exemplo, arquivos simples. O diagrama a seguir mostra uma solução de sincronização utilizando Conectores nos modos Agente Iterativo e Delta. O Conector do Agente Iterativo lê as entradas de uma origem de dados. As entradas lidas são comparadas com aquelas salvas no Armazenamento Delta na iteração anterior. O resultado da comparação é uma Entrada Delta designada com códigos de operação delta definindo a mudança feita – incluir/excluir/modificar. Então a Entrada Delta é utilizada pelo Conector no modo Delta para aplicar as mudanças detectadas em um sistema de destino.

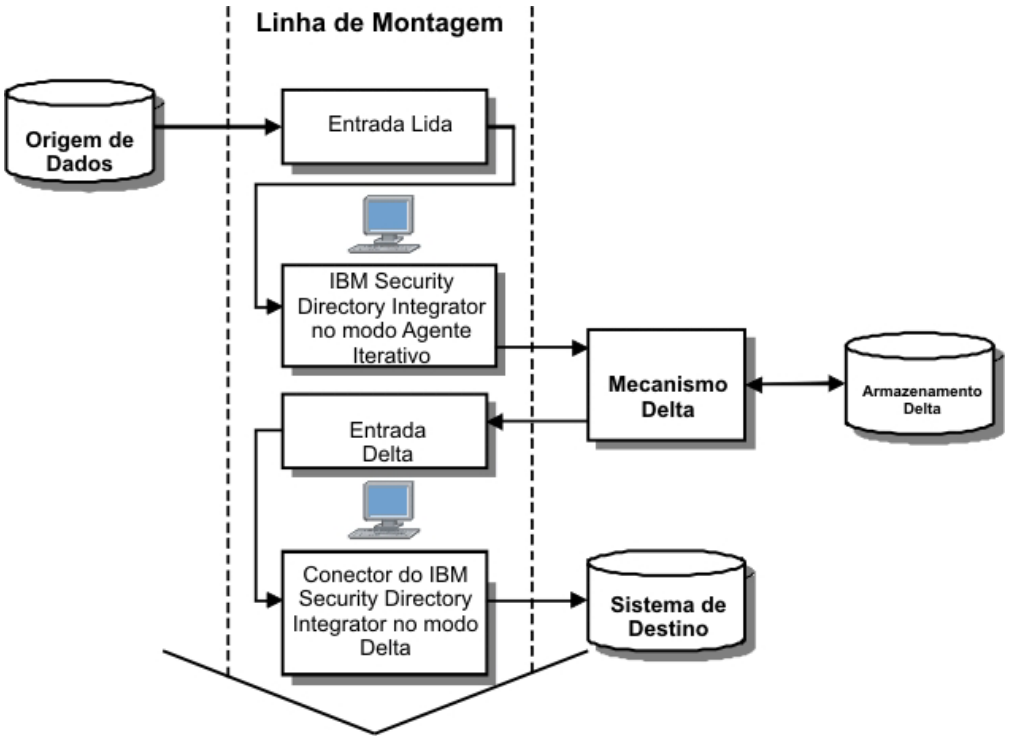


Figura 117. Sincronização AssemblyLine Utilizando Funcionalidade Delta

O resultado da execução do AssemblyLine é que os dados contidos na origem de dados são sincronizados com os dados no sistema de destino

Entradas Delta e Modo Atualizar

Os conectores no modo Atualizar têm um recurso de controle de mudança adicional chamado "Alterações de Cálculo". Esse recurso pode trabalhar com Entradas Delta.

O recurso "Alterações de Cálculo" pode ser ativado em um Conector no modo Atualizar através da seleção de uma caixa de opção desse nome na área de janela Detalhes do Conector. Quando ativado, o Conector compara sua Entrada com a Entrada real no sistema conectado. Se as duas Entradas forem iguais, não será feita nenhuma atualização pelo Conector. Portanto, a opção "Alterações de Cálculo" permite que o Conector no modo Atualizar ignore atualizações desnecessárias. Isso é útil para origens de dados com operações de atualização relativamente pesadas.

Entretanto, quando um Conector no modo Atualizar com o parâmetro **Alterações de Cálculo** recebe uma Entrada Delta, a lógica "Alterações de Cálculo" não é acionada; ele utiliza os códigos de operação delta designados à Entrada Delta para aplicar as atualizações necessárias no sistema conectado.

Exemplos

Exemplos de Delta Simples

As instruções abaixo mostrarão como utilizar alguns dos recursos Delta discutidos acima.

Configure um Conector do Sistema de Arquivos no recurso do mecanismo Delta ativado. Faça com que ele seja repetido em um documento XML simples que possa ser facilmente modificado em um editor de texto. Por exemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<DocRoot>
    <Entry>
        <Telephone>
            <ValueTag>111-1111</ValueTag>
            <ValueTag>222-2222</ValueTag>
            <ValueTag>333-3333</ValueTag>
        </Telephone>
        <Birthdate>1958-12-24</Birthdate>
        <Title>Full-Time SDI Specialist</Title>
        <uid>jdoe</uid>
        <FullName>John Doe</FullName>
    </Entry>
</DocRoot>
```

Assegure-se de usar o caractere especial de mapa de atributo para mapear tudo, o asterisco (*). Esse é o único Atributo necessário no mapa para garantir que todos os Atributos retornados sejam mapeados para o objeto de Entrada de Trabalho.

Agora, inclua um Componente de Script com o seguinte código:

```
// Get the names of all Attributes in work as a String array
var attName = work.getAttributeNames();

// Print the Entry-level delta op code
task.logmsg(" Entry (" +
work.getString( "FullName" ) + ") : " +
work.getOperation() );

// Loop through all the Attributes in work
for (i = 0; i < attName.length; i++) {

 // Grab an Attribute and print the Attribute-level op code
 att = work.getAttribute( attName[ i ] );
 task.logmsg("    Att ( " + attName[i] + ") : " + att.getOperation()
);

 // Now loop through all the Attribute's values and print their op codes
 for (j = 0; j < att.size(); j++) {
  task.logmsg( "      Val (" +
                 att.getValue( j ) + ") : " +
                 att.getValueOperation( j ) );
 }
}
```

Na primeira vez que você executar esse AL, o código do Componente de Script criará esta saída de log:

```
12:46:31   Entry
(John Doe) : add
12:46:31      Att ( Telephone) : replace
12:46:31        Val (111-1111) :
12:46:31        Val (222-2222) :
12:46:31        Val (333-3333) :
12:46:31      Att ( Birthdate) : replace
12:46:31        Val (1958-12-24) :
12:46:31      Att ( Title) : replace
12:46:31        Val (Full-Time SDI Specialist) :
12:46:31      Att ( uid) : replace
12:46:31        Val (jdoe) :
12:46:31      Att ( FullName) : replace
12:46:31        Val (John Doe) :
```

Como essa Entrada não foi encontrada no Armazenamento Delta anteriormente vazio, ela é ativada no nível de Entrada como nova. Além disso, cada um de seus Atributos possui um código replace, significando que todos os valores foram alterados (o que faz sentido, já que o Delta está informando que esses dados são novos).

Faça as seguintes alterações em seu arquivo XML:

1. Altere o último valor do número de Telephone para 333-4444.
2. Exclua Birthdate.
3. Inclua um novo Atributo Address.

Sua Configuração resultante deve ser semelhante a esta:

```
<?xml version="1.0" encoding="UTF-8"?>
<DocRoot>
    <Entry>
        <Telephone>
            <ValueTag>111-1111</ValueTag>
            <ValueTag>222-2222</ValueTag>
            <ValueTag>333-4444</ValueTag>
        </Telephone>
        <Title>Full-Time SDI Specialist</Title>
        <uid>jdoe</uid>
        <FullName>John Doe</FullName>
        <Address>123 Willowby Lane</Address>
    </Entry>
</DocRoot>
```

Execute novamente o AL. Dessa vez, a saída de log terá a seguinte aparência:

```
13:53:22   Entry (John Doe) : modify
13:53:22      Att ( Telephone) : modify
13:53:22        Val (111-1111) : unchanged
13:53:22        Val (222-2222) : unchanged
13:53:22        Val (333-4444) : add
13:53:22        Val (333-3333) : delete
13:53:22      Att ( Birthdate) : delete
13:53:22        Val (1958-12-24) : delete
13:53:22      Att ( uid) : unchanged
13:53:22        Val (jdoe) : unchanged
13:53:22      Att ( Title) : unchanged
13:53:22        Val (Full-Time SDI Specialist) : unchanged
13:53:22      Att ( Address) : add
13:53:22        Val (123 Willowby Lane) : add
13:53:22      Att ( FullName) : unchanged
13:53:22        Val (John Doe) : unchanged
```

Agora, a Entrada é rotulada como *modify* e os Atributos refletem as modificações de cada um deles. Como você pode ver, o Atributo Birthdate é marcado como *delete* e Address como *add*. Foi por isso que você utilizou o caractere especial de mapeamento completo para o Mapa de Entrada. Se você tivesse mapeado apenas os Atributos que existiam na primeira versão desse documento XML, não teríamos recuperado Address quando ele apareceu na entrada.

Preste atenção especial nas duas últimas entradas de valor no atributo Telephone marcado como *modify*. A mudança para um dos valores desse Atributo resultou em dois itens Delta: um valor *delete* e depois *add*.

Para construir um AssemblyLine de sincronização de dados em versões anteriores do IBM Security Directory Integrator, você tinha que criar um script para tratar o controle de fluxo. Embora você possa receber *adds*, *modifies* e *deletes* do recurso ou

componente alterado, um Conector só poderia ser configurado para um dos dois modos de saída necessários: Atualizar ou Excluir.

Portanto, você precisa de dois Conectores apontando para o mesmo sistema de destino e coloca um script no Gancho Antes de Executar de cada Conector, de forma a ignorar a Entrada se o seu código de operação não correspondesse ao modo desse componente. Ou, você poderia ter um único Conector (no modo Atualizar ou Excluir) no estado *Passivo* e, em seguida, controlar sua execução a partir do código de script, onde verificaria o código de operação. Isso ainda significaria que, mesmo sabendo o que foi alterado no caso de uma Entrada modificada, o Conector no Modo Atualizar ainda faria a leitura nos dados originais antes de gravar as alterações de volta na origem de dados. Isso pode levar a tráfego de rede ou de origem de dados indesejado quando você estiver apenas alterando um único valor em um Atributo relacionado a um grupo com diversos valores que contenha milhares de valores.

Mais Exemplos

Acesse o diretório `root_directory/examples/` da instalação do IBM Security Directory Integrator.

- No subdiretório `deltas`, você vai localizar uma configuração de IBM Security Directory Integrator simples que demonstra a funcionalidade Delta. Este demo é executado em sistemas MS Windows (Windows 2000) somente por usar o banco de dados MS Access e a ponte JDBC:ODBC. Se você utilizar outra plataforma, será necessário criar seu próprio banco de dados e configurar as definições de JDBC dos Conectores para esse banco de dados.
- No subdiretório `delta_tagging`, você vai localizar um exemplo que demonstra como converter uma Entrada identificada no nível do valor em uma Entrada regular e uma Entrada regular em uma Entrada Delta.

Capítulo 8. Painel do IBM Security Directory Integrator

Use o IBM Security Directory Integrator Dashboard para instalar, configurar, implementar e monitorar soluções de integração de dados.

Também é possível usar o Dashboard para criar e configurar soluções EasyETL (ETL significa Extrair, Transformar e Carregar) para integrações de dados simples. Uma solução EasyETL contém um único AssemblyLine com um conector de origem e destino.

Apresentando o Painel do IBM Security Directory Integrator

O Dashboard, um aplicativo da web, foi desenvolvido usando a estrutura Open Service Gateway Initiative (OSGI). Ele é usado para configurar e gerenciar soluções de integração de dados em um servidor por meio da interface RESTful (Representational State Transfer). A partir do Dashboard, é possível:

- Fazer upload de soluções de integração de dados existentes e instalá-las como uma solução normal ou salvar como um modelo
- Configurar soluções de integração de dados para refletir mudanças que são específicas de seu ambiente local
- Incluir planejamentos para executar AssemblyLines em horários especificados
- Construir AssemblyLines simples com um único conector de origem e de destino
- Navegar pelo conteúdo do conector selecionado
- Implementar, monitorar e auditar os processos de soluções de integração de dados
- Visualizar detalhes do servidor, log do servidor e dados de armazenamento do sistema
- Filtrar dados do arquivo de log configurando os critérios de filtro para extrair apenas as informações necessárias
- Visualizar o histórico de execução do AssemblyLine na forma de um gráfico
- Criar relatórios contendo dados no status de execução do AssemblyLine, enviado automaticamente para o usuário por meio de email

Vantagens do Painel do IBM Security Directory Integrator

As vantagens são:

- Facilita a implementação mais rápida e simples de soluções de integração de dados
- Cria e configura integrações simples sem usar o Configuration Editor
- Gera relatório de email planejado do histórico do AssemblyLine, enviado usando o recurso Dashboard RunReport. É possível usar o relatório para tratar de requisitos de alta disponibilidade/failover da implementação
- Facilita a integração com o ambiente de desenvolvimento do Eclipse
- Assegura um gerenciamento efetivo e eficiente de execuções do AssemblyLine

Acessando o Aplicativo Dashboard

É possível acessar o aplicativo da web Dashboard a partir do navegador ou do Configuration Editor.

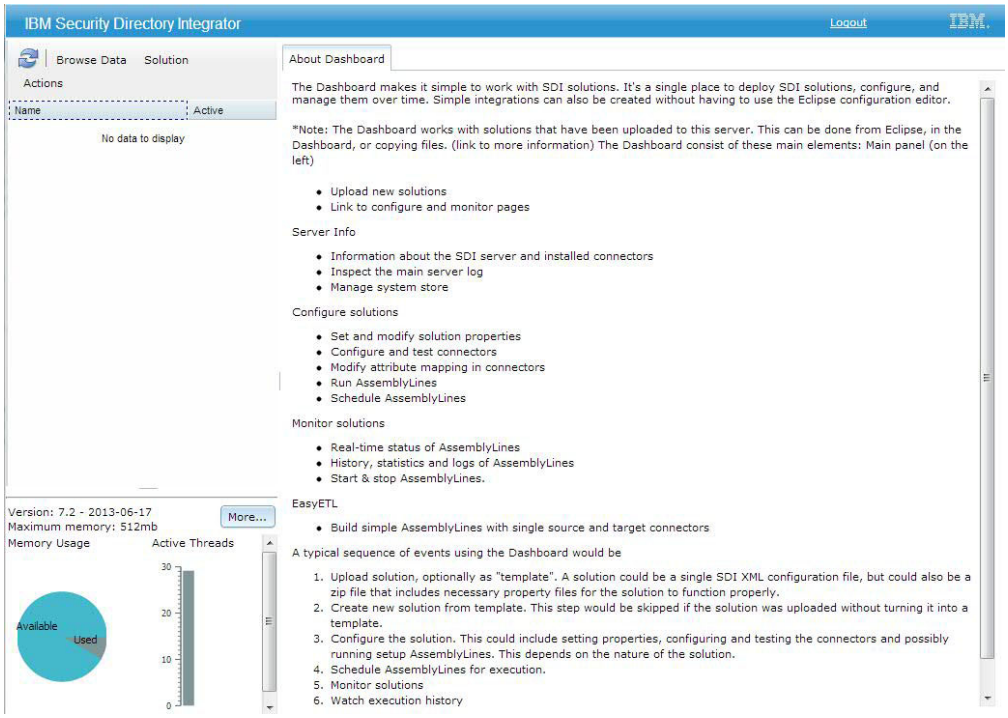
Antes de Iniciar

Inicie o IBM Security Directory Integrator Server antes de acessar o Dashboard.

Abrindo a partir de um Navegador

Procedimento

A partir de um navegador, acesse `https://<host name>:<rest-api-port>/dashboard`. A página inicial Painel é exibida.



Nota: Por padrão, é possível acessar o Dashboard no host local. Para autenticar um usuário, configure o arquivo de propriedades do IBM Security Directory Integrator para incluir o nome de usuário e a senha LDAP para os acessos remoto e local.

Abrindo a partir do Menu Iniciar do Windows

Procedimento

No menu **Iniciar** do Windows, escolha **Iniciar** > **Todos os Programas** > **IBM Security Directory Integrator v7.2** > **Abrir Painel**. A página inicial do IBM Security Directory Integrator Dashboard é exibida.

Tabela 14. Opções de Menu do Painel do IBM Security Directory Integrator

Opção de Menu	Descrição
Navegar por Dados	Use essa opção para configurar o conector selecionado e navegar por seu conteúdo.

Tabela 14. Opções de Menu do Painel do IBM Security Directory Integrator (continuação)

Opção de Menu		Descrição
Solução	**Monitorar**	Use essa opção para controlar o progresso de execução do AssemblyLine e visualizar o histórico de execução de sua solução.
	Iniciar	Use essa opção para iniciar a solução selecionada.
	Parar	Use essa opção para finalizar uma solução em execução.
	Configurar	Use essa opção para abrir a solução selecionada para modificação.
	Excluir	Use essa opção para excluir a solução selecionada do servidor.
	Desbloquear Solução	Use essa opção para desbloquear uma solução.
Ações	**Navegar por Dados**	Use essa opção para configurar o conector selecionado e navegar por seu conteúdo.
	Mostrar Log do Sistema	Use essa opção para visualizar o conteúdo do log do sistema (`ibmdi.log`).
	Criar Solução	Use essa opção para escolher um modelo e criar uma solução de integração de dados.
	Fazer Upload da Solução	Use essa opção para fazer upload de uma solução existente para o IBM Security Directory Integrator Server.
	Mostrar Detalhes do Servidor	Use essa opção para visualizar informações sobre o IBM Security Directory Integrator, como versão, componentes instalados e outras informações do servidor.

Configurações do Internet Explorer para acesso remoto

Incluir as definições de configuração necessárias para acessar o Painel a partir de um sistema remoto em um navegador Internet Explorer, em que o Internet Explorer Enhanced Security Configuration (IE ESC) esteja ativado.

Por padrão, o IE ESC bloqueia todos os scripts que estão em execução em uma página da web. O Painel carrega vários scripts antes de exibir qualquer coisa na página da web. Por essa razão, um navegador Internet Explorer ativado por IESC mostra uma página em branco quando você abre o Painel. Para acessar a página, você deve incluir sites que hospedam o Painel na lista segura do Internet Explorer.

1. No menu do **Internet Explorer**, clique em **Ferramentas** > **Opções de Internet**.
2. Clique na guia **Segurança**.
3. Clique em **Sites confiáveis**.
4. Clique em **Sites**.
5. No campo **Incluir este website à zona**, insira a URL do Painel. Por exemplo: `https://mydashboard.com/dashboard/*`.
6. Clique em **Incluir**.
7. Clique em **Fechar** e, em seguida, em **OK** para fechar a página e salvar as configurações.
8. Reinicie o navegador Internet Explorer.
9. Acesse o Painel no navegador Internet Explorer.

Fazendo Upload de uma Solução de Integração de Dados

Use a janela Fazer Upload de Soluções do Dashboard para fazer upload de uma solução de integração de dados existente para instalá-la no Servidor IBM Security Directory Integrator. Também é possível salvar a solução transferida por upload como um modelo.

Sobre Esta Tarefa

O Dashboard pode ser usado para fazer upload de uma solução de integração de dados existente do IBM Security Directory Integrator, que é um arquivo de configuração XML, para instalá-la no servidor. É possível modificar a solução com

as propriedades de requisito antes da implementação. Também é possível fazer upload da solução existente para salvá-la como um modelo. O modelo salvo pode ser usado para criar várias instâncias da mesma integração.

Procedimento

1. Na área de janela de navegação da janela Dashboard, clique em **Ações** > **Fazer Upload de Soluções**.
2. Defina as configurações a seguir na janela Fazer Upload de Soluções.

Opção	Descrição
Navegar	Navegue até a solução existente da qual deseja fazer upload
Fazer Upload como Modelo	Salva a solução transferida por upload como um modelo **Nota:** O modelo salvo não é instalado no Servidor IBM Security Directory Integrator (diretório de configuração).
Sobrescrever Solução Existente	Sobrescreve soluções instaladas existentes ou modelos
Nome da Solução	Especifica o nome da solução transferida por upload **Nota:** Também é possível visualizar nomes das soluções instaladas atualmente e das soluções de modelo na janela Fazer Upload de Soluções.

3. Clique em **OK**.

Criando uma Solução de Integração de Dados

Use a janela Criar Soluções do Dashboard para criar uma solução de integração de dados usando várias opções.

Procedimento

1. Na área de janela de navegação da janela Dashboard, clique em **Ações** > **Criar Soluções**.
2. Selecione uma das opções a seguir na janela Criar Soluções.

Opção	Descrição
Solução Instalada	A solução é criada com base em uma solução já instalada no Servidor IBM Security Directory Integrator.
Modelo	A solução é criada com base em uma solução existente, que é transferida por upload como um modelo.
Padrão	A solução EasyETL é criada tendo um único AssemblyLine com dois conectores.

3. Digite o nome de sua solução no campo **Nome da Solução**.
4. Clique em **OK**.

Configuração da Solução

O Dashboard consiste em vários editores para configurar os componentes da solução de integração de dados.

No Dashboard, é possível modificar a solução selecionada com configurações apropriadas que sejam específicas às suas necessidades antes da implementação. Para facilitar configurações mais rápidas, é possível limitar a quantia de informações visíveis durante a configuração de uma solução. Também é possível incluir novas páginas de configuração para incluir propriedades que sejam específicas à solução.

Apenas as soluções de integração de dados instaladas são mostradas no painel de navegação da janela Dashboard. Cada solução contém componentes, como AssemblyLines e conectores associados. Esses componentes são mostrados como uma estrutura em árvore no Solution Editor. A partir do Solution Editor é possível instalar, criar, configurar, implementar e monitorar suas soluções usando os seguintes editores:

- AssemblyLine Editor – esse editor é usado para criar e configurar planejamentos nos AssemblyLines e para executar o AssemblyLine com sua saída no visualizador de logs.
- Connector Editor – esse editor é usado para configurar os conectores do AssemblyLine.
- EasyETL Editor – esse editor é usado para configurar a solução EasyETL, que contém um único AssemblyLine com dois conectores.
- Monitor Editor – esse editor é usado para monitorar informações dos desempenhos atual e passado de sua solução.

O Solution Editor possui as seguintes opções de menu:

Opção	Descrição
Editar Interface da Solução	Use essa opção para selecionar os componentes da solução que ficam visíveis para edição.
Novo AssemblyLine	Use essa opção para criar uma solução EasyETL e incluí-la na solução selecionada.
Renomear AssemblyLine	Use essa opção para renomear o AssemblyLine selecionado.
Excluir AssemblyLine	Use essa opção para excluir AssemblyLines.
Criar RunReport	Use essa opção para criar RunReport para enviar email sobre os status das execuções de AssemblyLine.

Incluindo Descrição da Solução

Use o Solution Editor do Dashboard para incluir informações importantes sobre a solução de integração de dados.

Procedimento

1. Na área de janela de navegação da janela Dashboard, selecione a solução.
2. Clique em **Solução** > **Configurar**.
3. No Solution Editor, selecione **Descrição da Solução** e clique em **Editar Descrição**.
4. Digite a descrição na área de texto.
5. Clique em **Fechar**.
6. Clique em **Salvar**.

Configurando um Planejamento de AssemblyLine

Use o AssemblyLine Editor para criar ou configurar o planejador do Dashboard para executar AssemblyLines de sua solução de integração de dados no horário especificado.

Criando um Planejamento

Procedimento

1. Selecione o AssemblyLine na estrutura em árvore.
2. No AssemblyLine Editor, clique na guia **Planejamento**.
3. Clique em **Criar Planejamento**.
4. Defina as seguintes configurações:

Tabela 15. Opções de Planejamento

Configuração	Opção	Descrição
Planejamento	**Iniciar automaticamente quando a solução for iniciada**	Use essa opção para executar o AssemblyLine com base na opção de execução selecionada.
	Não iniciar se já estiver em execução	
	Finalizar o planejamento se o assemblyline falhar	
Mês	**Todos os Meses**	Use essa opção para selecionar os meses de execução do planejamento.
	Selecionar Meses	
Dia	**Todos os Dias**	Use essa opção para selecionar os dias de execução do planejamento.
	Dias Úteis	
	Selecionar Dias	
Horas/Minutos/Segundos	**Horas**	Use essa opção para selecionar a sincronização de execução do planejamento.
	Minutos	
	Segundos	
Compartilhar Criação de Log		Use essa opção para especificar se a criação de log entre o planejador e os AssemblyLines deve ser compartilhada ou não.

5. Clique em **Salvar**.

Excluindo um Planejamento

Procedimento

1. Selecione o AssemblyLine na estrutura em árvore.
2. No AssemblyLine Editor, clique na guia **Planejamento**.
3. Clique em **Excluir Planejamento**.
4. Clique em **Salvar**.

Executando e Parando o Planejador de Dashboard

Procedimento

1. Selecione o AssemblyLine na estrutura em árvore.
2. No AssemblyLine Editor, clique na guia **Executar**.
3. Para executar o AssemblyLine, clique em **Executar**. A saída é mostrada no visualizador de logs.
4. Para parar o AssemblyLine, clique em **Parar**.

Configurando um Conector

Use o Connector Editor para configurar detalhes do conector, como informações de mapeamento dos atributos e os detalhes de conexão, para adequar seus requisitos de configuração.

Modificando Detalhes da Conexão

Procedimento

1. Selecione o conector na estrutura em árvore de sua solução.
2. Clique na guia **Conector**.
3. Para visualizar apenas os campos importantes, clique em **Menos**.
4. Para alterar o tipo de conector, clique em **Escolher Componente** e selecione o conector na lista **Selecionar Conector**.
5. Dependendo dos requisitos, modifique as configurações de conexão.
6. Para testar a conexão com a origem de dados, clique em **Testar Conexão**.
7. Clique em **Salvar**.

Modificando o Mapeamento de Atributo

Procedimento

1. Selecione o conector na estrutura em árvore de sua solução.
2. Clique na guia **Mapa de Atributos**.
3. Para incluir um atributo:
   a. Clique em **Incluir**.
   b. Selecione um atributo de trabalho na lista ou digite um nome novo no arquivo de texto.
   c. Clique em **OK**.
4. Para ler dados do conector e mostrá-los no mapa de atributos, clique em **Ler Próximo**.
5. Para fechar a conexão com a origem de dados, clique em **Fechar Conexão**.
6. Para modificar o atributo selecionado, clique em **Ações**.
   a. Para editar a designação do atributo selecionado, clique em **Editar Atributo** e modifique o seguinte tipo de designação:
      - **Designação Simples** – é possível designar um atributo a partir da lista de atributos de origem.
      - **Designação com Script** – é possível gravar um script para o atributo na área de texto.
   b. Clique em **Fechar**.
   c. Para mapear o atributo selecionado, clique em **Mapear atributo**.
   d. Para remover o mapeamento do atributo selecionado, clique em **Remover Mapeamento do atributo**.
7. Clique em **Salvar**.

Dashboard EasyETL

O recurso Dashboard EasyETL pode ser usado para criar e configurar soluções de integração de dados simples.

O Dashboard EasyETL fornece uma interface com o usuário para construir e configurar AssemblyLines simples com um único conector de origem e de destino. Uma solução EasyETL pode ser:

- Planejada no Planejador do Dashboard
- Salva como um modelo
- Incluída em RunReports
- Implementada e monitorada no monitor de solução do Dashboard

Configurando Soluções EasyETL

Use o EasyETL Editor para configurar e incluir planejamentos em uma solução EasyETL.

Procedimento

1. Na área de janela de navegação da janela Dashboard, selecione a solução EasyETL.
2. Clique em **Solução** > **Configurar**.

   Alternativamente,

   a. Clique com o botão direito do mouse na solução selecionada.

   b. No menu, clique em **Configurar**.
3. No Solution Editor, selecione a solução EasyETL.
4. Inclua uma descrição para a solução. Consulte o tópico "Incluindo Descrição da Solução" na página 229 para obter mais detalhes.
5. No EasyETL Editor, selecione a solução e clique em **Configurar**.
6. Na lista de **Conectores**, selecione um componente para os conectores de origem e de destino.
7. Especifique os detalhes de configuração dos conectores selecionados no formulário.
8. Selecione um analisador na lista de **Analisadores**.

   Nota: A lista de **Analisadores** fica ativa apenas quando o conector selecionado requer ou pode usar um analisador.

   Para estabelecer a conexão e navegar pelos dados:

   a. Para estabelecer uma conexão e ler os 25 primeiros registros do conector, clique em **Conectar**.

   b. Para visualizar os próximos 25 registros, clique em **Avançar**.

   c. Para alternar a visualização de registro de dados para mostrar um registro de cada vez, clique em **Alternar Visualização**.

   d. Para visualizar os próximos registros na visualização alternada, clique em **Avançar**.

   e. Para visualizar apenas os atributos selecionados na tabela:

      1) Clique no ícone **Opções de Configuração** na barra de menus.

      2) Na janela Opções de Configuração, selecione os atributos.

      3) Para alterar a ordem dos atributos, clique nas setas para cima ou para baixo.

      4) Clique em **OK**.
9. Clique em **Voltar** para retornar para o EasyETL Editor.
10. Para visualizar os atributos dos conectores configurados, clique em **Descobrir**.
11. Para mapear atributos de origem selecionados para os atributos de destino, clique em **Mapear**. O seguinte tipo de mapeamento é criado com base na seleção do atributo de origem:

    - Se houver um único atributo selecionado na visualização de origem e também na visualização de destino, será criado um mapa de um para um entre os dois atributos.
    - Se houver diversos atributos selecionados na visualização de origem e nenhum selecionado na visualização de destino, será criado um mapa de um para um para cada atributo. Se não houver atributos na visualização de destino, eles serão criados.

- Se houver diversos atributos selecionados na visualização de origem e um único atributo selecionado na visualização de destino, execute uma das seguintes tarefas:
  - Clique em **Copiar** para copiar atributos de origem para seus atributos de destino equivalentes.
  - Clique em **Fundir** para fundir todos os valores da origem em um atributo no destino.
  - Clique em **Concatenar** para concatenar os valores do atributo de origem como um valor único no atributo de destino.

12. Clique em **Salvar**.

Configuração do Servidor

Use a janela Informações do Servidor para visualizar e modificar a configuração do servidor.

É possível visualizar as seguintes propriedades de servidor e configurar os comportamentos na janela Informações do Servidor:

- Especificando o número máximo de arquivos de log a serem criados e mantidos
- Iniciando e parando o Tombstone Manager
- Definindo configuração do Servidor LDAP para autenticar usuários para acessar o aplicativo Dashboard
- Visualizando informações do servidor, como a versão do IBM Security Directory Integrator com data de construção, nome do host, endereço IP, tempo de inicialização do servidor, nome do S.O. e ID do servidor
- Visualizando conectores e analisadores instalados no Servidor IBM Security Directory Integrator
- Visualizando conteúdo de vários armazenamentos do sistema que estão em uso pelo Servidor IBM Security Directory Integrator

No canto inferior esquerdo da janela Dashboard, é possível visualizar as informações da versão do aplicativo. Também é possível monitorar as seguintes métricas para obter uma captura instantânea de como o aplicativo está sendo executado:

- Gráfico de setores circulares de uso da memória
- Contagem de encadeamentos ativos para processo do servidor

Configurando Definições de Log

Use a guia Log e Tombstones para ativar ou desativar o criador de logs padrão e para especificar o número máximo de arquivos de log a serem salvos.

Procedimento

1. Na janela Dashboard, clique em **Ações** > **Mostrar Detalhes do Servidor** ou clique em **Mais**.
2. Clique na guia **Log e Tombstones**.
3. Defina as seguintes configurações:

Opção	Descrição
Criador de Logs Padrão	Ativa o criador de logs padrão **Nota:** Use o Criador de Logs Padrão, anexado a cada AssemblyLine que é executado, se desejar arquivos de log individuais para os AssemblyLines.
Número Máximo de Arquivos de Log	Especifica o número máximo de arquivos de log que pode ser salvo pelo criador de logs.

4. Clique em **Atualizar**.

Configurando Tombstones

Use a guia Log e Tombstones para iniciar o Dashboard Tombstone Manager, que cria registros de tombstone.

Sobre Esta Tarefa

O Tombstone Manager do Dashboard cria registros de tombstone para cada AssemblyLine assim que é finalizado. Um registro de tombstone contém estatísticas, como o número de entradas lidas por Agentes Iterativos, o número de registros ignorados e o número de registros atualizados. As estatísticas podem ser usadas para mostrar a carga de trabalho graficamente ao longo do tempo.

Procedimento

1. Na janela Dashboard, clique em **Ações** > **Mostrar Detalhes do Servidor** ou clique em **Mais**.
2. Para gerar os registros do tombstone, clique em **Iniciar o Tombstone Manager**.

   Nota: Para parar o Tombstone Manager, reinicie o Servidor IBM Security Directory Integrator.

Configurando Definições de Segurança do Dashboard

Use a guia Segurança e Conexão para configurar o Servidor LDAP para autenticar usuários para as conexões local e remota.

Procedimento

1. Na janela Dashboard, clique em **Ações** > **Mostrar Detalhes do Servidor** ou clique em **Mais**.
2. Clique na guia **Segurança e Conexão**.
3. Defina as seguintes configurações:

Opção	Descrição
Local	Especifica conexões do host local
Remoto	Especifica conexões do host não local
Nome do Host LDAP	Especifica o nome do Servidor LDAP
Base de Procura LDAP	Especifica o nome da base de procura LDAP para localizar usuários
DN do Grupo LDAP	Especifica o nome do DN do Grupo LDAP para verificar a associação dos usuários

4. Clique em **Atualizar**.

Visualizando Componentes Instalados

Use a guia Componentes Instalados para visualizar os componentes, como conectores e analisadores, que são instalados no Servidor IBM Security Directory Integrator.

Procedimento

1. Na janela Dashboard, clique em **Ações** > **Mostrar Detalhes do Servidor** ou clique em **Mais**.
2. Para visualizar a lista de componentes, clique na guia **Componentes Instalados**.

Visualizando Dados de Armazenamento do Sistema

Use a guia Armazenamentos do Servidor para visualizar o conteúdo dos vários armazenamentos do sistema que estão em uso pelo Servidor IBM Security Directory Integrator.

Procedimento

1. Na janela Dashboard, clique em **Ações** > **Mostrar Detalhes do Servidor** ou clique em **Mais**.
2. Para visualizar a lista de armazenamentos do servidor, clique na guia **Armazenamentos do Servidor**.

Dashboard RunReports

Use o recurso Dashboard RunReport para criar relatórios de email automatizados do histórico de execução do AssemblyLine.

Você cria um AssemblyLine que é executado com base nos planejamentos configurados para gerar relatórios de email automatizados para os AssemblyLines monitorados. O RunReport AssemblyLine usa o banco de dados de tombstones do IBM Security Directory Integrator para determinar se os AssemblyLines foram executados desde a última vez que o RunReport foi executado.

O relatório de email contém informações sobre o histórico de execução que indica se os AssemblyLines monitorados foram ou não executados com sucesso. Esse relatório é enviado periodicamente por correio para o destinatário especificado.

Criando RunReports

Use a janela Criar RunReport para criar um RunReport AssemblyLine. Esse AssemblyLine é usado para gerar relatórios de email automatizados. Os relatórios de email contêm informações sobre o histórico de execução dos AssemblyLines monitorados e são enviados periodicamente para os destinatários especificados.

Criando e Planejando um RunReport

Procedimento

1. No Solution Editor, clique em **Ações** > **Criar RunReport**.
2. Na janela Criar RunReport, digite um nome para o RunReport AssemblyLine no campo **Nome**.
3. Clique em **OK**.
4. No AssemblyLine Editor, defina as seguintes configurações:

Tabela 16. Opções de Planejamento de RunReport

Configuração	Opção	Descrição
Planejamento	**Iniciar automaticamente quando a solução for iniciada**	Executa o AssemblyLine com base na opção de execução selecionada
	Não iniciar se já estiver em execução	
	Finalizar o planejamento se o assemblyline falhar	
Mês	**Todos os Meses**	Especifica os meses de execução do planejamento
	Mês(es) Selecionado(s)	
Dia	**Todos os Dias**	Especifica os dias de execução do planejamento
	Dias Úteis	
	Dia(s) Selecionado(s)	
Horas/Minutos/Segundos	**Horas**	Especifica a sincronização de execução do planejamento
	Minutos	
	Segundos	
Assunto (sucesso)		Especifica a linha de assunto do email para execução bem-sucedida dos AssemblyLines monitorados **Nota:** O relatório gerado usa a linha de assunto com base na execução bem-sucedida ou não dos AssemblyLines monitorados, desde o último relatório gerado.
Assunto (falha)		Especifica a linha de assunto do email para indicar execução com falha dos AssemblyLines monitorados
Endereço de Remetente		Especifica o endereço de email do remetente
Endereço do Destinatário		Especifica o endereço de email do destinatário **Nota:** Especifique uma lista delimitada por vírgula ou ponto e vírgula para diversos endereços de email.
Host SMTP		Parâmetro de host SMTP que aceita conexões SMTP e transfere correio para o destinatário
Monitorar AssemblyLines		Uma lista separada por vírgula de nomes de AssemblyLine para serem monitorados
Compartilhar Criação de Log		Especifica se a criação de log entre o planejador e os AssemblyLines deve ser compartilhada ou não

5. Clique em **Salvar**.

Excluindo um Planejamento
Procedimento

1. Selecione o RunReport AssemblyLine na estrutura em árvore.
2. No AssemblyLine Editor, clique na guia **Planejamento**.
3. Clique em **Excluir Planejamento**.
4. Clique em **Salvar**.

Executando e Parando o Planejador de RunReport
Procedimento

1. Selecione o RunReport AssemblyLine na estrutura em árvore.
2. No AssemblyLine Editor, clique na guia **Executar**.
3. Para executar o AssemblyLine, clique em **Executar**. A saída é mostrada no visualizador de logs.
4. Para parar o AssemblyLine, clique em **Parar**.

Configurando e Navegando pelos Dados do Conector

Use a página Navegar por Dados no Solution Editor para configurar um conector, navegar por origens de dados do conector ou para criar uma solução de integração de dados.

Procedimento

1. Na área de janela de navegação da janela Dashboard, clique em **Navegar por Dados**.
2. Na janela Navegar por Dados, na lista de **Conectores**, selecione um conector para configurar e navegar por seu conteúdo.
3. Selecione um analisador na lista de **Analisadores**.

   Nota: A lista de **Analisadores** fica ativa apenas quando o conector selecionado requer ou pode usar um analisador.
4. Configure os detalhes do conector no formulário do lado direito da janela.
5. Para criar uma solução para o conector selecionado, clique em **Criar Integração**.
6. Para estabelecer uma conexão e ler os 25 primeiros registros do conector, clique em **Conectar**.
7. Para visualizar os próximos 25 registros, clique em **Avançar**.
8. Para alternar a visualização de registro de dados para mostrar um registro de cada vez, clique em **Alternar Visualização**.
9. Para visualizar os próximos registros na visualização alternada, clique em **Avançar**.
10. Para visualizar apenas os atributos selecionados na tabela:
    a. Clique no ícone **Opções de Configuração** na janela Navegar por Dados.
    b. Na janela Opções de Configuração, selecione os atributos.
    c. Para alterar a ordem dos atributos, clique nas setas para cima ou para baixo.
    d. Clique em **OK**.

Monitor de Solução

Use o Dashboard Monitor Editor para controlar o progresso de execução de AssemblyLine e visualizar o histórico de execução de sua solução.

É possível visualizar AssemblyLines da solução selecionada no Monitor Editor, que apresenta informações dos desempenhos atual e passado dos AssemblyLines. É possível monitorar as seguintes atividades:

- Monitoramento do status em tempo real dos AssemblyLines, que inclui:
  - Horário de início da execução do AssemblyLine
  - Horário de encerramento da execução do AssemblyLine
  - Horário planejado da próxima execução do AssemblyLine
  - Número de objetos de dados que são processados nos AssemblyLines
- Histórico de execução dos AssemblyLines, que são mostrados graficamente como carga de trabalho ao longo do tempo
- Arquivos de log para gravar resultados e problemas de cada execução de AssemblyLine
- Registros de tombstones de todos os AssemblyLines na solução
- Arquivo de log do servidor (`ibmdi.log`)

Iniciando e Parando os AssemblyLines

Use o Dashboard Monitor Editor para iniciar e parar os AssemblyLines.

Procedimento

1. No Dashboard, selecione a solução instalada.
2. Clique em **Solução** > **Monitorar**.

   Alternativamente,

   a. Clique com o botão direito do mouse na solução selecionada.
   b. No menu, clique em **Monitorar**.
3. No Monitor Editor, selecione o AssemblyLine para executar as seguintes ações:
   - Para iniciar um AssemblyLine, clique em **Iniciar**.
   - Para parar um AssemblyLine, clique em **Parar**.

   Os detalhes de execução são mostrados no editor. O ícone verde com o nome AssemblyLine indica que o AssemblyLine está ativo.

Visualizando o Histórico de Execução do AssemblyLine

Use o Dashboard Monitor Editor para visualizar o histórico de execução do AssemblyLine na forma de um gráfico.

Procedimento

1. No Monitor Editor, selecione o AssemblyLine.
2. Clique em **Histórico** ou clique duas vezes no AssemblyLine selecionado. Os detalhes de execução do AssemblyLine são mostrados em uma visualização gráfica como carga de trabalho ao longo do tempo.
3. Selecione um dos seguintes contadores para representar em gráfico:
   - Get
   - Erros
   - Incluir
   - Excluir
   - Consultar
   - Modificar
4. Clique no nó no gráfico para visualizar o arquivo de log de uma execução específica. Também é possível visualizar as estatísticas de execução dessa execução como uma dica de ferramenta.

Visualizando Registros de Tombstone

Use a guia Tombstones no Dashboard Monitor Editor para visualizar os registros de tombstone criados quando a execução de AssemblyLines for concluída.

Procedimento

1. Acesse o Monitor Editor do Dashboard.
2. Para visualizar os tombstones registrados, clique na guia **Tombstones**.

   Nota: É possível visualizar os registros de tombstone apenas quando o Tombstone Manager está em execução. Para obter mais informações, consulte o tópico "Configurando Tombstones" na página 234.

Visualizando Arquivos de Log

Use a guia Arquivos de Log no Dashboard Monitor Editor para visualizar os arquivos de log criados para cada execução de AssemblyLine. Os arquivos de log são usados para analisar problemas rápida e facilmente e para depurar qualquer problema.

Procedimento

1. No Monitor Editor, clique na guia **Arquivos de Log**. Os arquivos de log de toda a execução de AssemblyLine são mostrados como uma estrutura em árvore.
2. Para visualizar o conteúdo do arquivo de log, selecione o arquivo de log na estrutura em árvore e clique duas vezes.
3. Para procurar informações específicas no conteúdo do log, digite o texto que você deseja procurar no campo **Procurar**.
4. Selecione a caixa de seleção **Incluir Origem** na origem da mensagem ou no componente que registrou a mensagem.
5. Clique em **Opções** para configurar os seguintes detalhes:

Opção	Descrição
Tamanho da Página	
Tipos de Mensagens	Selecione um dos seguintes tipos de mensagens para ser incluído no arquivo de log: • INFO • WARN • ERROR • DEBUG
Opções de Exibição	Selecione opções para mostrar data, horário, origem da mensagem ou números de linha no arquivo de log: • Mostrar Data • Mostrar Horário • Incluir Origem • Mostrar Números da Linha

Avisos

Estas informações foram desenvolvidas para produtos e serviços oferecidos nos Estados Unidos. A IBM pode não oferecer os produtos, serviços ou recursos discutidos neste documento em outros países. Consulte um representante IBM local para obter informações sobre produtos e serviços disponíveis atualmente em sua área. Qualquer referência a produtos, programas ou serviços IBM não significa que apenas produtos, programas ou serviços IBM possam ser utilizados. Qualquer produto, programa ou serviço funcionalmente equivalente, que não infrinja nenhum direito de propriedade intelectual da IBM poderá se utilizado em substituição a este produto, programa ou serviço. Entretanto, a avaliação e verificação da operação de qualquer produto, programa ou serviço não IBM são de responsabilidade do Cliente.

A IBM pode ter patentes ou solicitações de patentes pendentes relativas a assuntos tratados nesta publicação. O fornecimento desta publicação não lhe garante direito algum sobre tais patentes. Pedidos de licença devem ser enviados, por escrito, para:

Gerência de Relações Comerciais e Industriais da IBM Brasil
Av. Pasteur, 138-146
Botafogo
Rio de Janeiro, RJ
CEP 22290-240

Para consultas sobre licenças a respeito de informações do conjunto de caracteres de byte duplo (DBCS), entre em contato com o Departamento de Propriedade Intelectual da IBM em seu país ou envie consultas, por escrito, para:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tóquio 103-8510, Japão

O parágrafo a seguir não se aplica a nenhum país em que tais disposições não estejam de acordo com a legislação local:

A INTERNATIONAL BUSINESS MACHINES CORPORATION FORNECE ESTA PUBLICAÇÃO "NO ESTADO EM QUE SE ENCONTRA", SEM GARANTIA DE NENHUM TIPO, SEJA EXPRESSA OU IMPLÍCITA, INCLUINDO, MAS A ELAS NÃO SE LIMITANDO, AS GARANTIAS IMPLÍCITAS DE NÃO INFRAÇÃO, COMERCIALIZAÇÃO OU ADEQUAÇÃO A UM DETERMINADO PROPÓSITO.

Alguns países não permitem a exclusão de garantias expressas ou implícitas em certas transações; portanto, essa disposição pode não se aplicar ao Cliente.

Essas informações podem conter imprecisões técnicas ou erros tipográficos. São feitas alterações periódicas nas informações aqui contidas; tais alterações serão incorporadas em futuras edições desta publicação. A IBM pode, a qualquer momento, aperfeiçoar e/ou alterar os produtos e/ou programas descritos nesta publicação, sem aviso prévio.

Referências nestas informações a Web sites não IBM são fornecidas apenas por conveniência e não representam de forma alguma um endosso a esses Web sites. Os materiais desses Web sites não fazem parte dos materiais desse produto IBM e a utilização desses Web sites é de inteira responsabilidade do Cliente.

A IBM pode utilizar ou distribuir as informações fornecidas da forma que julgar apropriada sem incorrer em qualquer obrigação para com o Cliente.

Licenciados deste programa que desejam obter informações sobre este assunto com objetivo de permitir: (i) a troca de informações entre programas criados independentemente e outros programas (incluindo este) e (ii) a utilização mútua das informações trocadas, devem entrar em contato com:

Gerência de Relações Comerciais e Industriais da IBM Brasil
Av. Pasteur, 138-146
Botafogo
Rio de Janeiro, RJ
CEP 22290-240

Tais informações podem estar disponíveis, sujeitas a termos e condições apropriadas, incluindo em alguns casos o pagamento de uma taxa.

O programa licenciado descrito nesta publicação e todo o material licenciado disponível são fornecidos pela IBM sob os termos do Contrato com o Cliente IBM, do Contrato Internacional de Licença do Programa IBM ou de qualquer outro contrato equivalente.

Todos os dados de desempenho aqui contidos foram determinados em um ambiente controlado. Portanto, os resultados obtidos em outros ambientes operacionais podem variar significativamente. Algumas medidas podem ter sido tomadas em sistemas em nível de desenvolvimento e não há garantia de que estas medidas serão iguais em sistemas geralmente disponíveis. Além disso, algumas medidas podem ter sido estimadas por extrapolação. Os resultados reais podem variar. Os usuários deste documento devem verificar os dados aplicáveis para seu ambiente específico.

As informações relativas a produtos não IBM foram obtidas junto aos fornecedores dos respectivos produtos, de seus anúncios publicados ou de outras fontes disponíveis publicamente. A IBM não testou estes produtos e não pode confirmar a precisão de seu desempenho, compatibilidade nem qualquer outra reivindicação relacionada a produtos não IBM. Dúvidas sobre os recursos de produtos não IBM devem ser encaminhadas diretamente a seus fornecedores.

Todas as declarações relacionadas aos objetivos e intentos futuros da IBM estão sujeitas a mudanças ou retirada sem aviso e representam apenas metas e objetivos.

Todos os preços IBM mostrados são preços de varejo sugeridos pela IBM, são atuais e estão sujeitos a alteração sem aviso prévio. Os preços do revendedor podem variar.

Estas informações foram projetadas apenas com o propósito de planejamento. As informações aqui contidas estão sujeitas a alterações antes que os produtos descritos estejam disponíveis.

Estas informações contêm exemplos de dados e relatórios utilizados nas operações diárias de negócios. Para ilustrá-los da forma mais completa possível, os exemplos

incluem nomes de indivíduos, empresas, marcas e produtos. Todos estes nomes são fictícios e qualquer semelhança com nomes e endereços utilizados por uma empresa real é mera coincidência.

LICENÇA DE COPYRIGHT:

Estas informações contêm programas de aplicativos de amostra na linguagem fonte, ilustrando as técnicas de programação em diversas plataformas operacionais. O Cliente pode copiar, modificar e distribuir estes programas de amostra sem a necessidade de pagar à IBM, com objetivos de desenvolvimento, utilização, marketing ou distribuição de programas aplicativos em conformidade com a interface de programação de aplicativo para a plataforma operacional para a qual os programas de amostra são criados. Esses exemplos não foram testados completamente em todas as condições. Portanto, a IBM não pode garantir ou implicar a confiabilidade, manutenção ou função destes programas. O Cliente pode copiar, modificar e distribuir estes programas de amostra de qualquer maneira sem pagamento à IBM, com objetivos de desenvolvimento, uso, marketing ou distribuição de programas de aplicativos em conformidade com interfaces de programação de aplicativos da IBM.

Cada cópia ou parte destes programas de amostra ou qualquer trabalho derivado deve incluir um aviso de copyright com os dizeres:

© (nome da empresa) (ano). Partes deste código são derivadas dos Programas de Amostra da IBM Corp. © Copyright IBM Corp. _digite o ano ou anos_. Todos os direitos reservados.

Se estas informações estiverem sendo exibidas em cópia eletrônica, as fotografias e ilustrações coloridas podem não aparecer.

Marcas Registradas

IBM, o logotipo IBM e ibm.com são marcas ou marcas registradas da International Business Machines Corp., registradas em várias jurisdições no mundo inteiro. Outros nomes de produtos e serviços podem ser marcas registradas da IBM ou de outras empresas. Uma lista atual de marcas registradas da IBM está disponível na web em "Copyright and trademark information" em www.ibm.com/legal/copytrade.shtml.

Adobe, Acrobat, PostScript e todas as marcas registradas baseadas no Adobe são marcas ou marcas registradas da Adobe Systems Incorporated nos Estados Unidos e/ou em outros países.

IT Infrastructure Library é uma marca registrada da Agência Central de Computação e Telecomunicações que agora faz parte do Departamento de Comércio do Governo.

Intel, o logotipo Intel, Intel Inside, o logotipo Intel Inside, Intel Centrino, o logotipo Intel Centrino, Celeron, Intel Xeon, Intel SpeedStep, Itanium e Pentium são marcas ou marcas registradas da Intel Corporation ou suas subsidiárias nos Estados Unidos e outros países.

Linux é uma marca registrada da Linus Torvalds nos Estados Unidos e/ou em outros países.

Microsoft, Windows, Windows NT e o logotipo Windows são marcas registradas da Microsoft Corporation nos Estados Unidos e/ou em outros países.

ITIL é uma marca registrada e uma marca registrada da comunidade do Departamento de comércio do governo e está registrada no Departamento de Marcas e Patentes dos Estados Unidos.

UNIX é uma marca registrada do The Open Group nos Estados Unidos e em outros países.

Java e todas as marcas registradas e logotipos baseados em Java são marcas ou marcas registradas da Oracle e/ou de suas afiliadas.

Cell Broadband Engine é uma marca registrada da Sony Computer Entertainment, Inc. nos Estados Unidos e/ou em outros países e é usada sob licença.

Linear Tape-Open, LTO, o Logotipo LTO, Ultrium e o logotipo Ultrium são marcas registradas da HP, IBM Corp. e Quantum nos Estados Unidos e em outros países.

Índice Remissivo

J

L

M

N

O

P

R

S

T

U

V

W

Impresso no Brasil

S517-9705-03