

---

Artigo

[Angelo Bruno Braga](#) · Mar. 2, 2021 9min de leitura

[Open Exchange](#)

## Exposição de classes persistentes do Plain ObjectScript como sistemas de código FHIR e conjuntos de valores

A especificação do [FHIR Terminology Service](#) descreve um conjunto de operações nos recursos [CodeSystem](#), [ValueSet](#) e [ConceptMap](#). Entre essas operações, as quatro operações a seguir parecem ser as mais amplamente adotadas:

CodeSystem

[\\$lookup](#)

[\\$validate-code](#)

ValueSet

[\\$expand](#)

[\\$validate-code](#)

O desenvolvimento de uma implementação parcial da especificação tem sido uma forma eficaz de explorar o [novo framework FHIR](#) introduzido no IRIS for Health 2020.1. A [implementação](#) inclui quatro operações listadas acima e oferece suporte a interações de [leitura](#) e [pesquisa](#) para os recursos [CodeSystem](#) e [ValueSet](#).

É importante observar que a implementação usa classes persistentes do Plain ObjectScript como fonte para tabelas de terminologia.

### Instalação e teste com uma estratégia de exemplo

A lista a seguir descreve as etapas básicas de instalação e teste:

1. Instale o IRIS for Health 2020.1 ou mais recente.
2. Configure um novo namespace usando o menu System Administration > Configuration > System Configuration > Namespaces no Portal ou executando o comando do `##class(HS.HC.Util.Installer).InstallFoundation("<name for the new namespace>")` no namespace HSLIB. Em seguida, importe as classes das pastas [src/cls](#) e [samples/cls](#) do repositório [intersystems-ru/fhir-terminology-service](#) no GitHub.
3. Crie um conjunto customizado de metadados FHIR com base no conjunto R4 com parâmetros de pesquisa adicionais definidos em [dummy-search-parameters.json](#). Isso pode ser feito usando o utilitário interativo `##class(HS.FHIRServer.ConsoleSetup).Setup()` ou executando o comando:

```
do ##class(HS.FHIRServer.Installer).InstallMetadataSet("", "", "HL7v40", $lb("<directory with dummy-search-parameters.json>"), 1)
```

- Esta etapa é necessária para que as operações `$expand` e `$validate-code` suportem solicitações HTTP GET.
  - Observe que os arquivos do conjunto de metadados FHIR empacotados com InterSystems IRIS estão no diretório `<installation directory>/dev/fhir/fhir-metadata`.
4. Crie um novo endpoint FHIR com base no novo conjunto de metadados e na classe [Sample.iscru.fhir.fts.SimpleStrategy](#). Novamente, isso pode ser feito usando o utilitário interativo ou executando o comando:

```
do ##class(HS.FHIRServer.Installer).InstallInstance("<web app URI, e.g. /csp/terminology>", "Sample.iscru.fhir.fts.SimpleStrategy", "")
```

5. Permita o acesso não autenticado ao novo endpoint: use o utilitário interativo ou execute os seguintes

comandos:

```
set strategy = ##class(HS.FHIRServer.API.InteractionsStrategy).GetStrategyForEndpoint
("<web app URI>")
set config = strategy.GetServiceConfigData()
set config.DebugMode = 4
do strategy.SaveServiceConfigData(config)
```

6. Popule o [Sample.iscru.fhir.fts.model.CodeTable](#):

```
do ##class(Sample.iscru.fhir.fts.model.CodeTable).Populate(10)
```

7. Importe o arquivo [fhir-terminology-service.postmancollection.json](#) para o Postman, ajuste a url variável definida na coleção e teste o serviço em `Sample.iscru.fhir.fts.model.CodeTable`, que é uma classe persistente simples.

## Interações FHIR suportadas

Atualmente, o único parâmetro de pesquisa com suporte para `CodeSystem` e `ValueSet` é url.

Os métodos HTTP GET e HTTP POST são suportados para as quatro operações listadas acima.

A tabela a seguir lista algumas das possíveis solicitações HTTP GET na classe

[Sample.iscru.fhir.fts.model.CodeTable](#).

URI	Descrição
(a ser acrescentado com <code>http://&lt;server&gt;:&lt;port&gt;&lt;web app URI&gt;</code> )	
/metadata	Obtém o recurso de declaração de capacidade do endpoint.
/CodeSystem/Sample.iscru.fhir.fts.model.CodeTable	Lê o recurso <code>CodeSystem</code> correspondente à classe <code>Sample.iscru.fhir.fts.model.CodeTable</code> .
/ValueSet/Sample.iscru.fhir.fts.model.CodeTable	Lê o recurso <code>ValueSet</code> correspondente à classe <code>Sample.iscru.fhir.fts.model.CodeTable</code> .
/CodeSystem?url=urn:CodeSystem:CodeTable	Pesquisa o recurso <code>CodeSystem</code> por url.
/CodeSystem	Envia todos os recursos do <code>CodeSystem</code> disponíveis.
/ValueSet?url=urn:ValueSet:CodeTable	Pesquisa o recurso <code>ValueSet</code> por url.
/ValueSet	Envia todos os recursos do <code>ValueSet</code> disponíveis.
/CodeSystem/\$lookup?system=urn:CodeSystem:CodeTable&code=TEST	Dado o sistema e o código, obtêm todos os detalhes sobre o conceito.
/ValueSet/\$expand?url=urn:ValueSet:CodeTable	Expand o <code>ValueSet</code> .
/CodeSystem/Sample.iscru.fhir.fts.model.CodeTable/\$validate-code?code=TEST	Valida se um código está no sistema de código.

## Criando uma estratégia personalizada

A fim de expor suas próprias classes persistentes como sistemas de código/conjuntos de valores FHIR, você precisará criar sua classe de estratégia personalizada ao subclassificar o [iscru.fhir.fts.FTSStrategy](#) e, em seguida, criar um endpoint FHIR com base na nova estratégia personalizada (veja a instalação no passo 4 acima).

Um parâmetro de classe e pelo menos três métodos devem ser substituídos por sua classe de estratégia:

- O parâmetro de classe `StrategyKey` deve receber algum valor único. O nome da classe atual parece ser uma boa opção.
- O método de classe `getCodeTablePackage()` deve retornar o nome do pacote para um determinado sistema de código (ou conjunto de valores) identificado por sua [URL canônica](#). Normalmente, todas as classes de terminologia pertencem a um pacote, portanto, esse método normalmente retornaria um e o

mesmo nome de pacote, independentemente dos valores dos argumentos.

- Os métodos de classe `getCodePropertyName()` e `getDisplayPropertyName()` devem retornar nomes de propriedades de classe que correspondem aos elementos de conceito `code` e `display`. Classes diferentes podem ter propriedades diferentes mapeadas para elementos de terminologia `code/display`.

Outros métodos e parâmetros de [iscru.fhir.fts.FTSStrategy](#) que você pode achar apropriado substituir são os seguintes:

- O método da classe `listCodeTableClasses()` precisa ser sobrescrito para dar suporte às solicitações de pesquisa que resultam no retorno de todos os sistemas de código disponíveis (ou conjuntos de valores). Este método deve retornar uma lista de nomes de classes de todas as classes de terminologia disponíveis. [Sample.iscru.fhir.fts.SimpleStrategy](#) contém a seguinte implementação básica deste método:

```
/// Retorna uma lista de todas as classes de tabela de código disponíveis.
ClassMethod listCodeTableClasses() As %List
{
    #dim sql As %String = "SELECT name FROM %Dictionary.ClassDefinition WHERE name LI
    KE ' " _ ..#codeTablePACKAGE _ ".%" ORDER BY name"
    #dim resultSet As %SQL.StatementResult = ##class(%SQL.Statement).%ExecDirect(, sq
    l)
    if (resultSet.%SQLCODE '= 0) && (resultSet.%SQLCODE '= 100) $$$ThrowStatus($$$$ERR
    OR($$$$SQLError, resultSet.%SQLCODE, resultSet.%Message))

    #dim return As %List = ""
    while resultSet.%Next()
    {
        set return = return _ $lb(resultSet.name)
    }

    quit return
}
```

- O método da classe `isExcludedProperty()` deve ser substituído se alguma propriedade particular de suas classes persistentes não aparecer nos recursos `CodeSystem`. Por padrão, este método filtra as propriedades de `Collection`, `Identity`, `Internal`, `MultiDimensional` e `Private`. Observe que a referência de objeto e as propriedades de fluxo atualmente não são suportadas e são ignoradas pelo framework.
- Os parâmetros de classe `codeSystemUrlPREFIX` e `valueSetUrlPREFIX`, e os métodos `getCodeSystemForClassname()`, `getValueSetForClassname()`, `determineCodeTableClassname()` e `determineCodeSystemForValueSet()` controlam como os nomes de classe são mapeados para [URLs canônicas](#) e vice-versa. Por padrão, o seguinte esquema de nomenclatura é usado para URLs canônicas:  

CodeSystem	ValueSet
urn:CodeSystem:<short class name>	urn:ValueSet:<short class name>

Observe que o [logical id](#) (também conhecido como server id) de um recurso `CodeSystem/ValueSet` é igual ao nome completo de sua classe correspondente.

## A FAZER

Atualmente falta suporte para controle de versão de sistemas de código, hierarquias de conceito e operação de `$subsumes`, recurso de `ConceptMap` e muitas outras coisas. Ideias e pull requests são bem-vindos!

[#FHIR #InterSystems IRIS for Health](#)

[Confira o aplicativo relacionado no InterSystems Open Exchange](#)

---

URL de  
origem: <https://pt.community.intersystems.com/post/exposi%C3%A7%C3%A3o-de-classes-persistentes-do-plain-objectscript-como-sistemas-de-c%C3%B3digo-fhir-e>