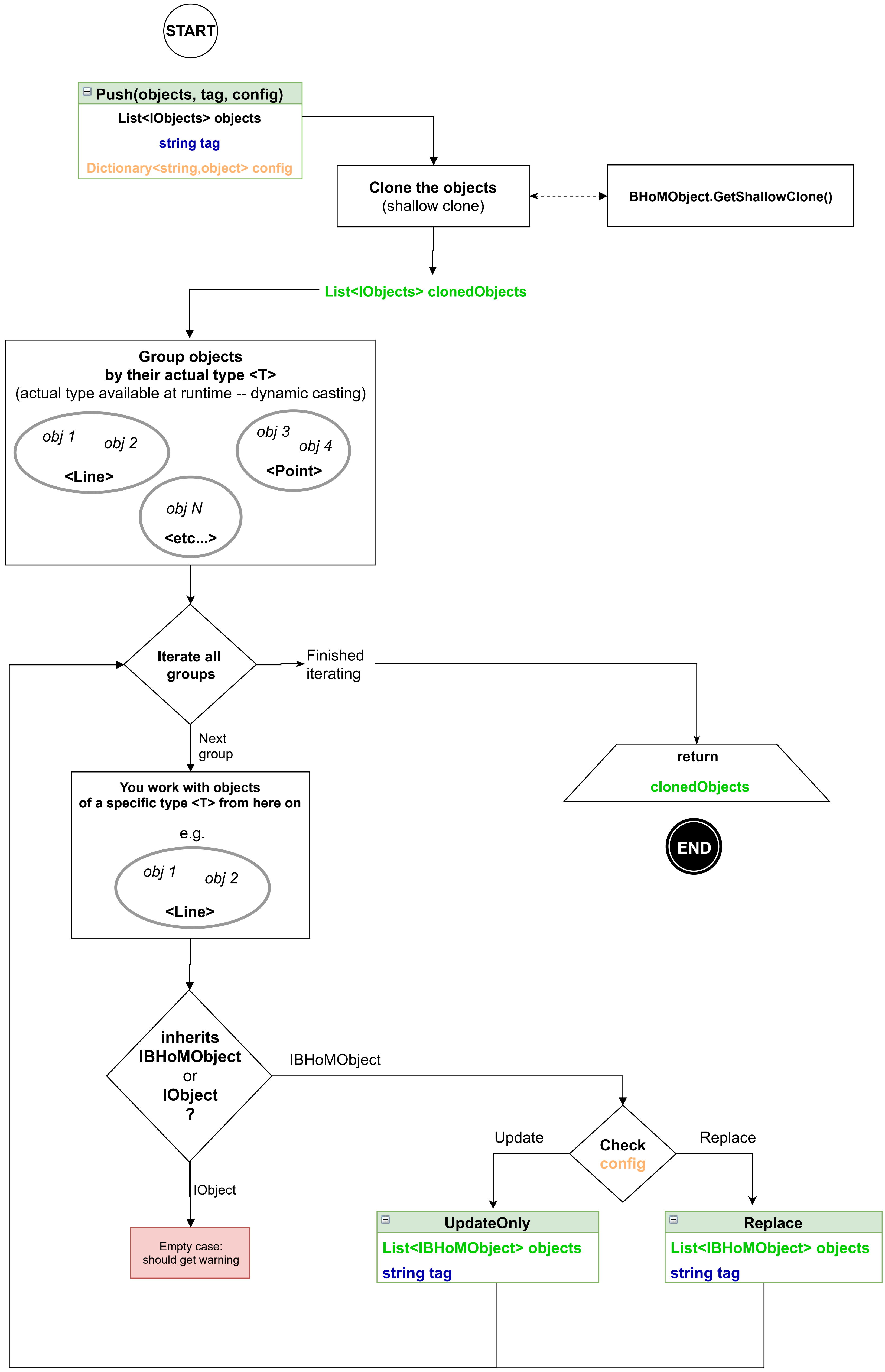
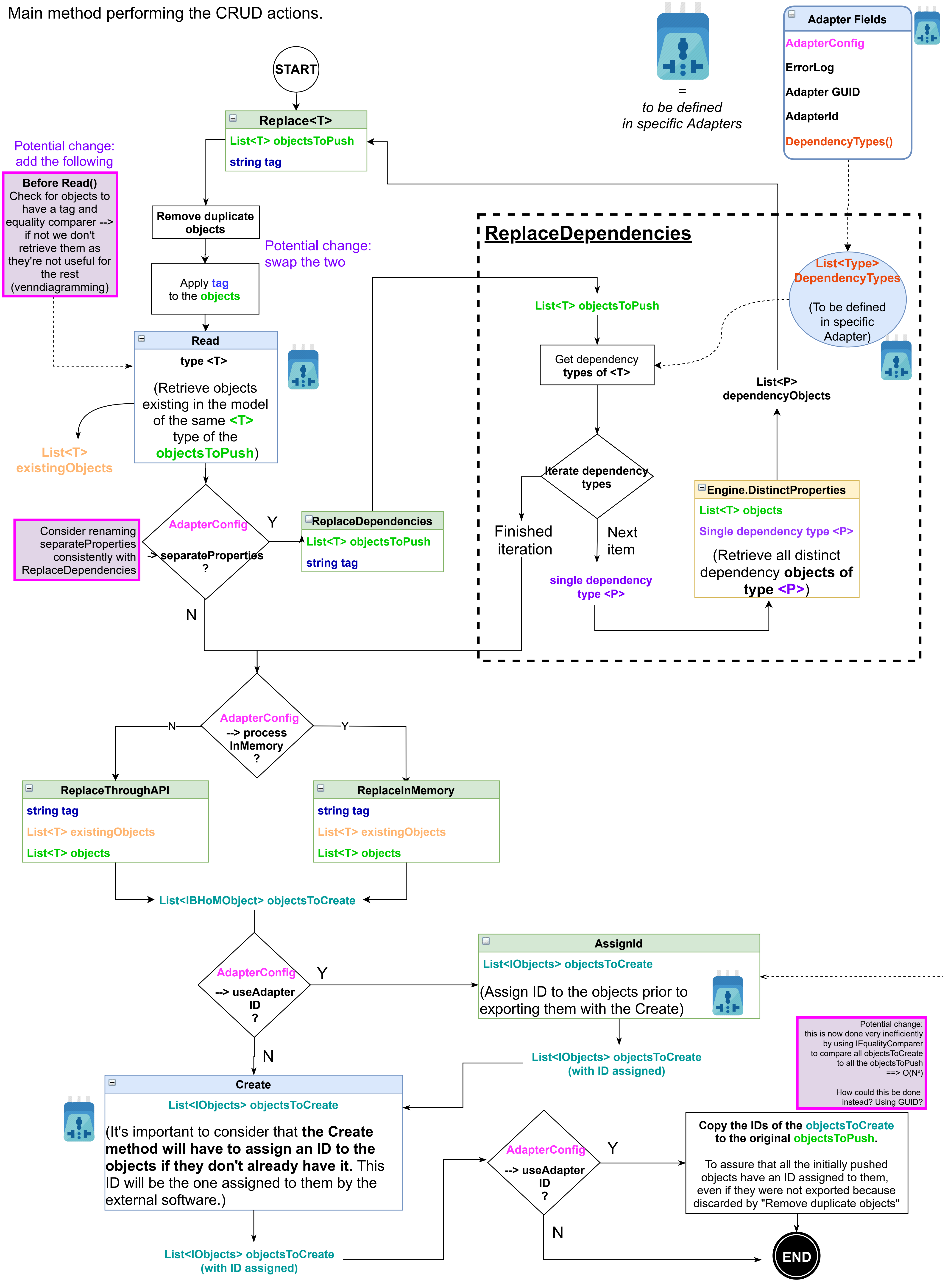
PUSH

Main method performing the CRUD actions.
Can be overridden by adapter implementations.



Replace

Main method performing the CRUD actions.



= to be defined in specific Adapters

Adapter Fields
AdapterConfig
ErrorLog
Adapter GUID
AdapterId
DependencyTypes()

START

Replace<T>
List<T> objectsToPush
string tag

Potential change: add the following

Before Read() Check for objects to have a tag and equality comparer --> if we don't retrieve them as they're not useful for the rest (venndiagramming)

Remove duplicate objects

Potential change: swap the two

Apply tag to the objects

Read
type <T>
(Retrieve objects existing in the model of the same <T> type of the objectsToPush)

List<T> existingObjects

Consider renaming separateProperties consistently with ReplaceDependencies

AdapterConfig -> separateProperties ?

Y

N

ReplaceDependencies
List<T> objectsToPush
string tag

ReplaceDependencies

List<T> objectsToPush

Get dependency types of <T>

List<Type> DependencyTypes
(To be defined in specific Adapter)

Iterate dependency types

Finished iteration

Next item

single dependency type <P>

Engine.DistinctProperties
List<T> objects
Single dependency type <P>
(Retrieve all distinct dependency objects of type <P>)

List<P> dependencyObjects

AdapterConfig --> process InMemory ?

N

Y

ReplaceThroughAPI
string tag
List<T> existingObjects
List<T> objects

ReplaceInMemory
string tag
List<T> existingObjects
List<T> objects

List<IBHoMObject> objectsToCreate

AdapterConfig --> useAdapter ID ?

Y

N

AssignId
List<IObjects> objectsToCreate
(Assign ID to the objects prior to exporting them with the Create)

List<IObjects> objectsToCreate (with ID assigned)

Potential change: this is now done very inefficiently by using IEqualityComparer to compare all objectsToCreate to all the objectsToPush ==> O(N²)

How could this be done instead? Using GUID?

Create
List<IObjects> objectsToCreate
(It's important to consider that **the Create method will have to assign an ID to the objects if they don't already have it**. This ID will be the one assigned to them by the external software.)

List<IObjects> objectsToCreate (with ID assigned)

AdapterConfig --> useAdapter ID ?

Y

N

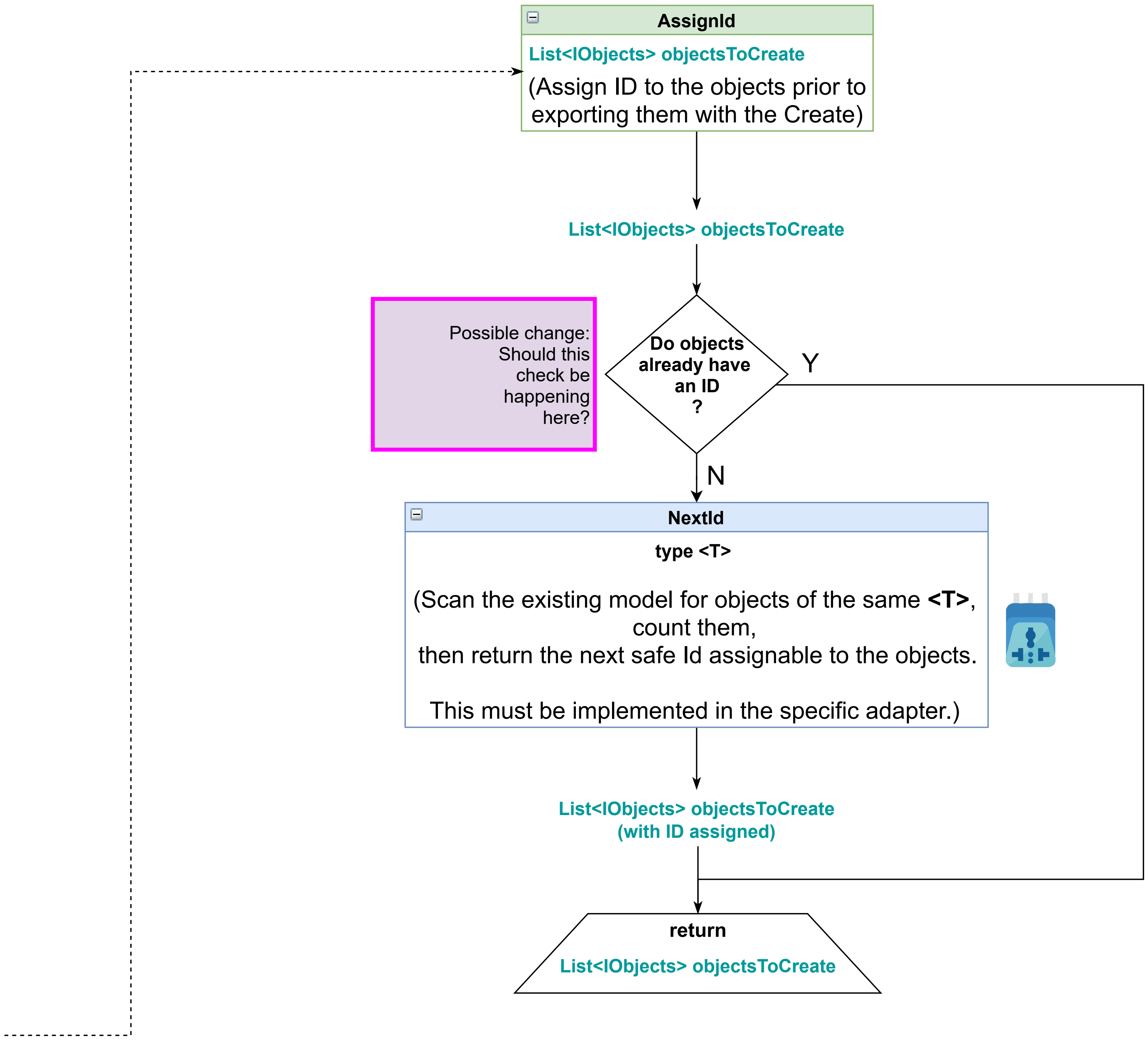
Copy the IDs of the objectsToCreate to the original objectsToPush.

To assure that all the initially pushed objects have an ID assigned to them, even if they were not exported because discarded by "Remove duplicate objects"

END

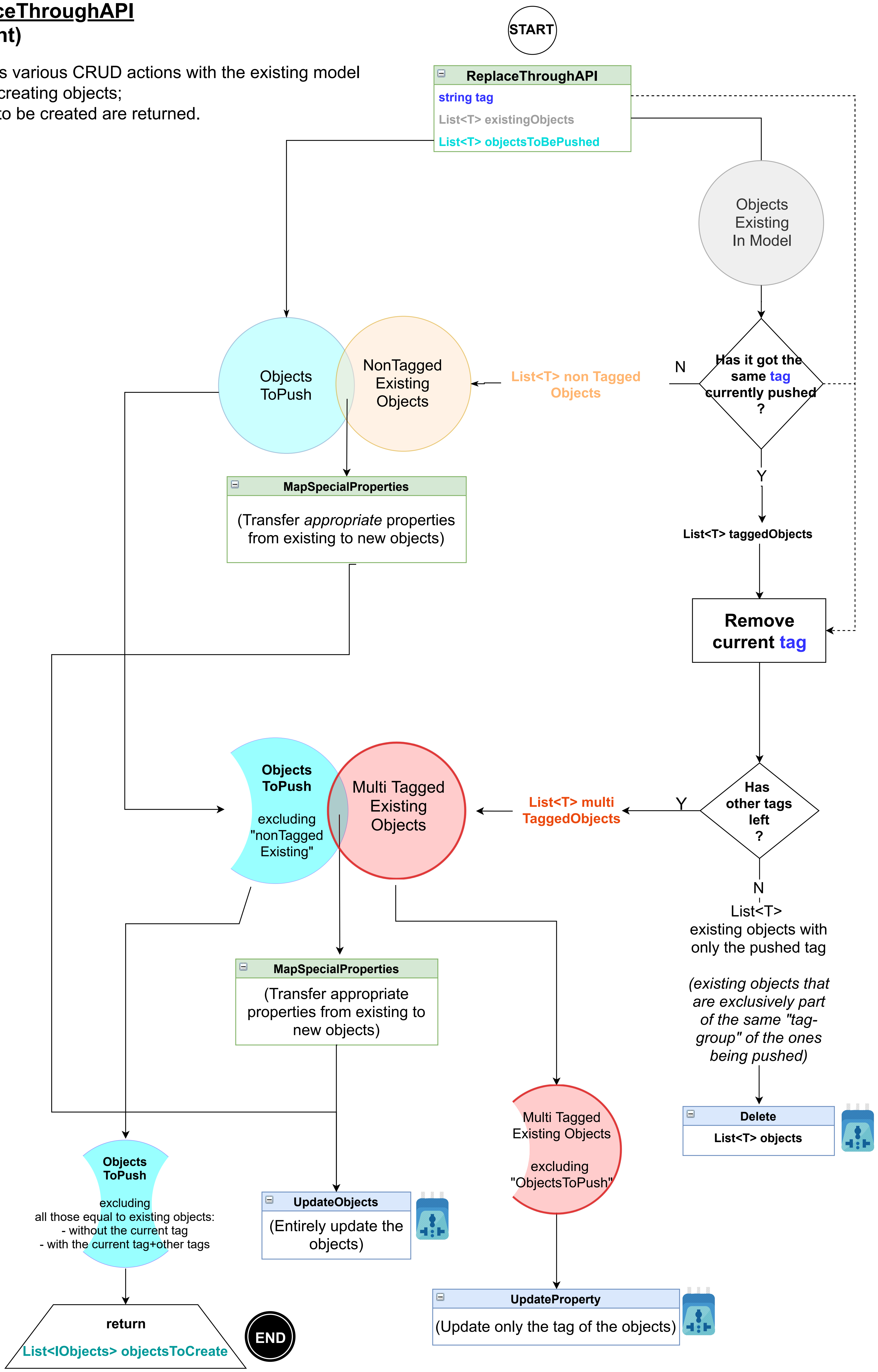
AssignId method

This method assigns a safe ID to the objects before attempting to export them.


AssignId
List<IObjects> objectsToCreate
(Assign ID to the objects prior to exporting them with the Create)
List<IObjects> objectsToCreate
Possible change: Should this check be happening here?
Do objects already have an ID ?
Y
N
NextId
type <T>
(Scan the existing model for objects of the same <T>, count them,
then return the next safe Id assignable to the objects.
This must be implemented in the specific adapter.)
List<IObjects> objectsToCreate (with ID assigned)
return
List<IObjects> objectsToCreate

ReplaceThroughAPI
(current)

Performs various CRUD actions with the existing model **except** creating objects;
objects to be created are returned.



START

ReplaceThroughAPI
string tag
List<T> existingObjects
List<T> objectsToBePushed

Objects Existing In Model

Has it got the same tag currently pushed ?

N

List<T> non Tagged Objects

Y

List<T> taggedObjects

Objects ToPush

NonTagged Existing Objects

MapSpecialProperties
(Transfer *appropriate* properties from existing to new objects)

Remove current tag

Has other tags left ?

Y

List<T> multi TaggedObjects

N

List<T>
existing objects with only the pushed tag

(existing objects that are exclusively part of the same "tag-group" of the ones being pushed)

Delete
List<T> objects

Objects ToPush
excluding "nonTagged Existing"

Multi Tagged Existing Objects

MapSpecialProperties
(Transfer appropriate properties from existing to new objects)

Multi Tagged Existing Objects
excluding "ObjectsToPush"

UpdateProperty
(Update only the tag of the objects)

UpdateObjects
(Entirely update the objects)

Objects ToPush
excluding
all those equal to existing objects:
- without the current tag
- with the current tag+other tags

return
List<IObjects> objectsToCreate

END

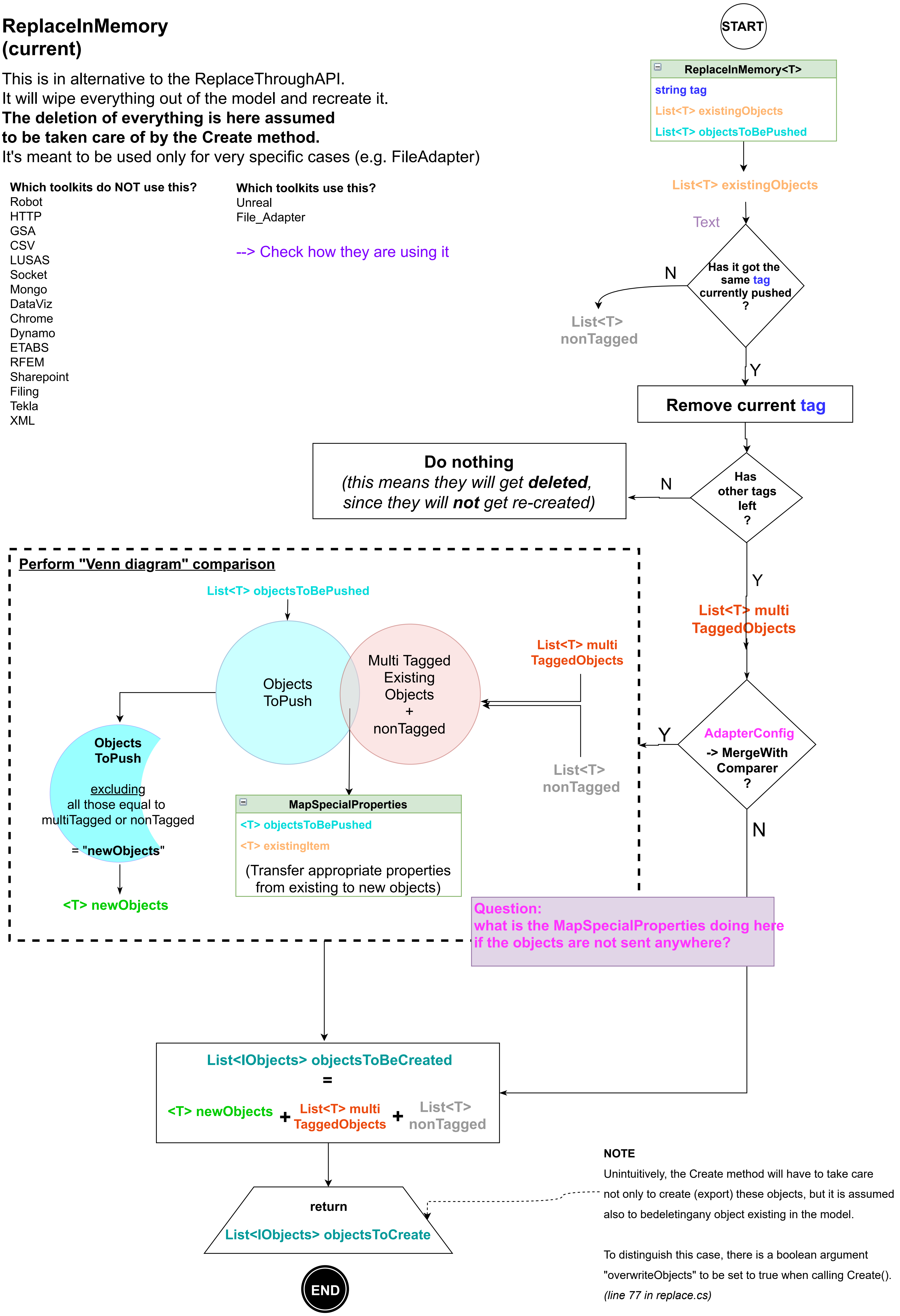
ReplaceInMemory (current)

This is in alternative to the ReplaceThroughAPI.
It will wipe everything out of the model and recreate it.
The deletion of everything is here assumed to be taken care of by the Create method.
It's meant to be used only for very specific cases (e.g. FileAdapter)

Which toolkits do NOT use this?
Robot
HTTP
GSA
CSV
LUSAS
Socket
Mongo
DataViz
Chrome
Dynamo
ETABS
RFEM
Sharepoint
Filing
Tekla
XML

Which toolkits use this?
Unreal
File_Adapter

--> Check how they are using it



START

ReplaceInMemory<T>
string tag
List<T> existingObjects
List<T> objectsToBePushed

List<T> existingObjects

Text

Has it got the same tag currently pushed ?

N

List<T> nonTagged

Y

Remove current tag

Has other tags left ?

N

Do nothing
*(this means they will get **deleted**, since they will **not** get re-created)*

Y

List<T> multi TaggedObjects

AdapterConfig -> MergeWith Comparer ?

Y

N

Perform "Venn diagram" comparison

List<T> objectsToBePushed

Objects ToPush

Multi Tagged Existing Objects + nonTagged

List<T> multi TaggedObjects

List<T> nonTagged

Objects ToPush

excluding all those equal to multiTagged or nonTagged

= "**newObjects**"

<T> newObjects

MapSpecialProperties
<T> objectsToBePushed
<T> existingItem
(Transfer appropriate properties from existing to new objects)

Question:
what is the MapSpecialProperties doing here if the objects are not sent anywhere?

List<IObjects> objectsToBeCreated
=
<T> newObjects + List<T> multi TaggedObjects + List<T> nonTagged

return
List<IObjects> objectsToCreate

END

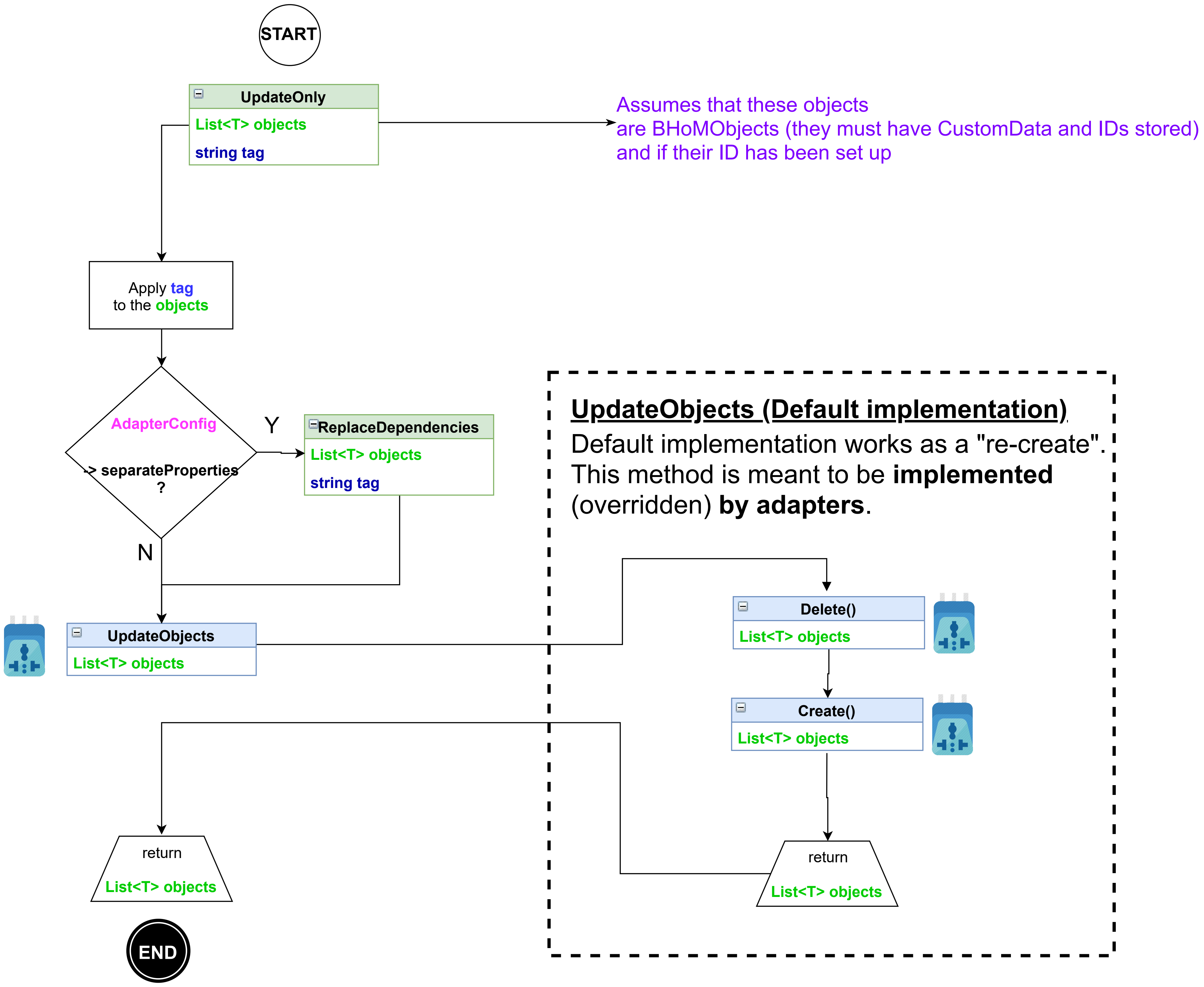
NOTE

Unintuitively, the Create method will have to take care not only to create (export) these objects, but it is assumed also to bedeletingany object existing in the model.

To distinguish this case, there is a boolean argument "overwriteObjects" to be set to true when calling Create().
(line 77 in replace.cs)

UpdateOnly

Adapter wrapper of the concrete UpdateObjects method



START

UpdateOnly

List<T> objects

string tag

Assumes that these objects are BHoMObjects (they must have CustomData and IDs stored) and if their ID has been set up

Apply tag to the objects

AdapterConfig -> separateProperties ?

Y

ReplaceDependencies

List<T> objects

string tag

N

UpdateObjects

List<T> objects

return List<T> objects

END

UpdateObjects (Default implementation)

Default implementation works as a "re-create". This method is meant to be **implemented** (overridden) **by adapters**.

Delete()

List<T> objects

Create()

List<T> objects

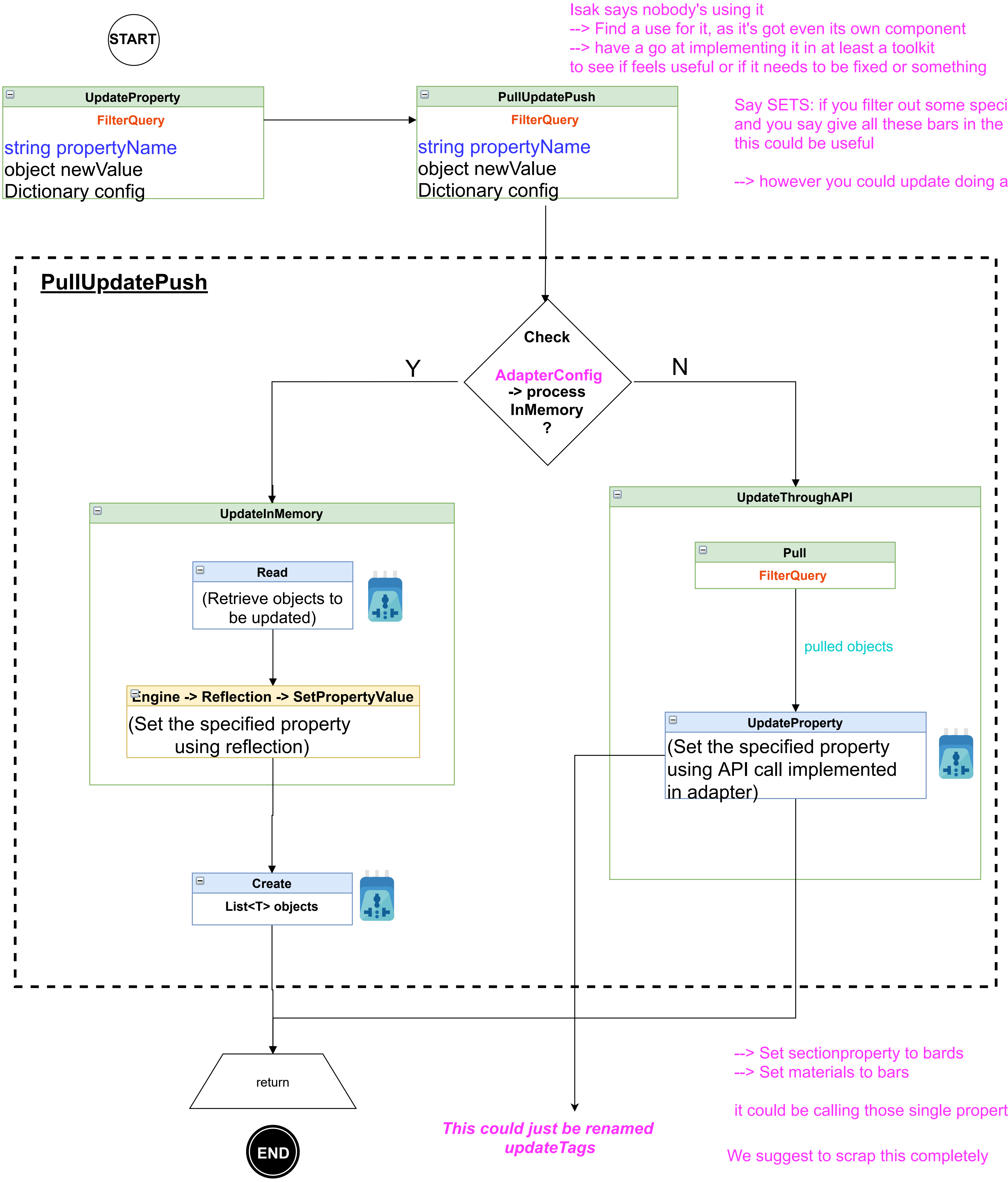
return List<T> objects

UPDATEPROPERTY
(Public Adapter Method)

Default implementation of the method commonly used to update a single property of specified objects (e.g. only the tag).

This method can be overridden by adapters.
Not to be confused with the UpdateProperty() Protected Abstract CRUD Method.

Lack of implementation --> check if anyone is using this
Isak says nobody's using it
--> Find a use for it, as it's got even its own component
--> have a go at implementing it in at least a toolkit
to see if feels useful or if it needs to be fixed or something



Say SETS: if you filter out some specific
and you say give all these bars in the E
this could be useful

--> however you could update doing a P

This could just be renamed updateTags

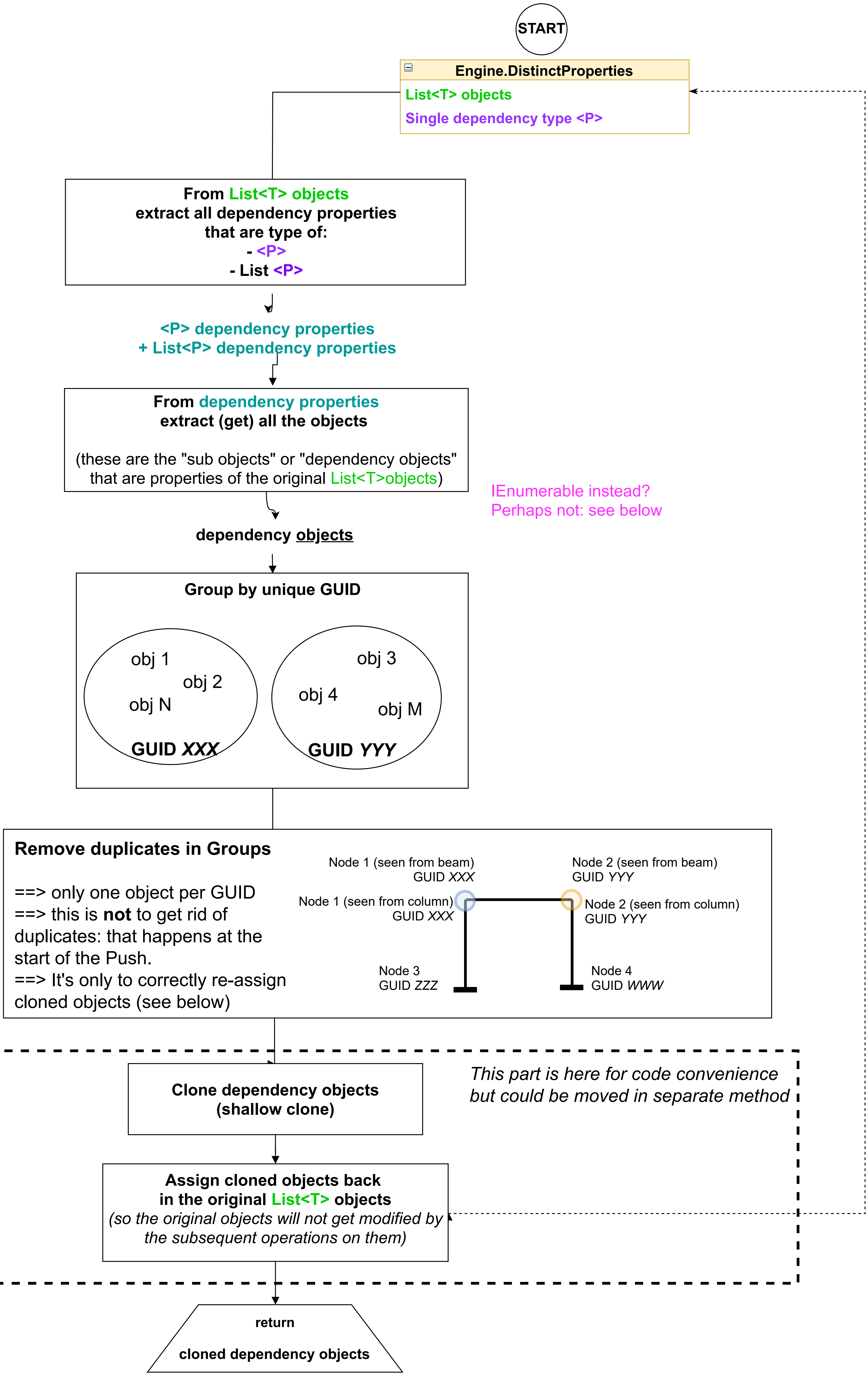
--> Set sectionproperty to bards
--> Set materials to bars

it could be calling those single propertie

We suggest to scrap this completely

Engine.DistinctProperties

Retrieves all distinct dependency properties of a specified type <P>
of the input objects of type <T>



START

Engine.DistinctProperties

List<T> objects

Single dependency type <P>

From List<T> objects
extract all dependency properties
that are type of:
- <P>
- List <P>

<P> dependency properties
+ List<P> dependency properties

From dependency properties
extract (get) all the objects

(these are the "sub objects" or "dependency objects"
that are properties of the original List<T>objects)

IEnumerable instead?
Perhaps not: see below

dependency objects

Group by unique GUID

obj 1
obj 2
obj N
GUID *XXX*

obj 3
obj 4
obj M
GUID *YYY*

Remove duplicates in Groups

==> only one object per GUID
==> this is **not** to get rid of duplicates: that happens at the start of the Push.
==> It's only to correctly re-assign cloned objects (see below)

Node 1 (seen from beam)
GUID *XXX*

Node 2 (seen from beam)
GUID *YYY*

Node 1 (seen from column)
GUID *XXX*

Node 2 (seen from column)
GUID *YYY*

Node 3
GUID *ZZZ*

Node 4
GUID *WWW*

Clone dependency objects
(shallow clone)

This part is here for code convenience
but could be moved in separate method

Assign cloned objects back
in the original List<T> objects
(so the original objects will not get modified by the subsequent operations on them)

return

cloned dependency objects

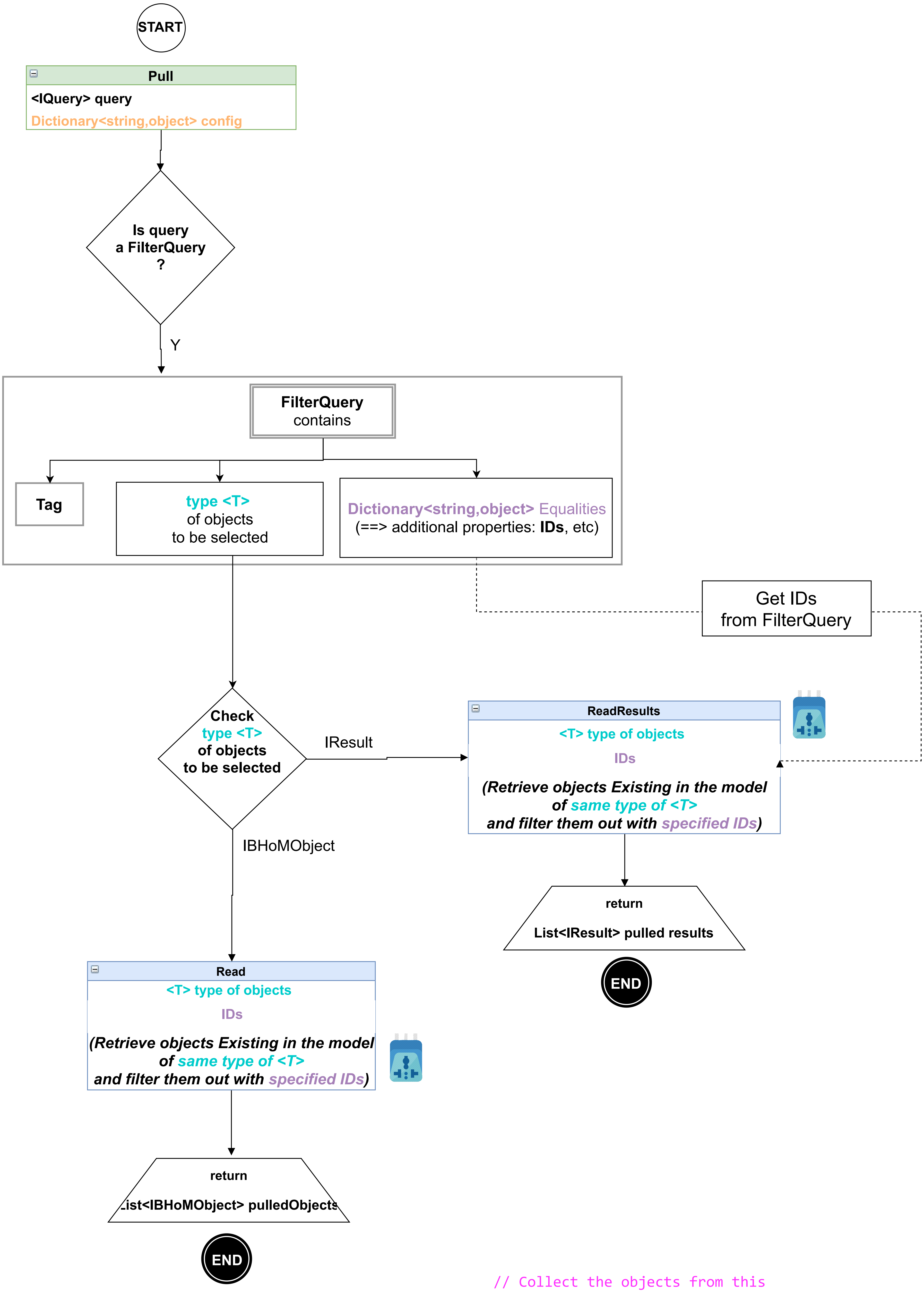
: tags
TABS

'USH

s

PULL

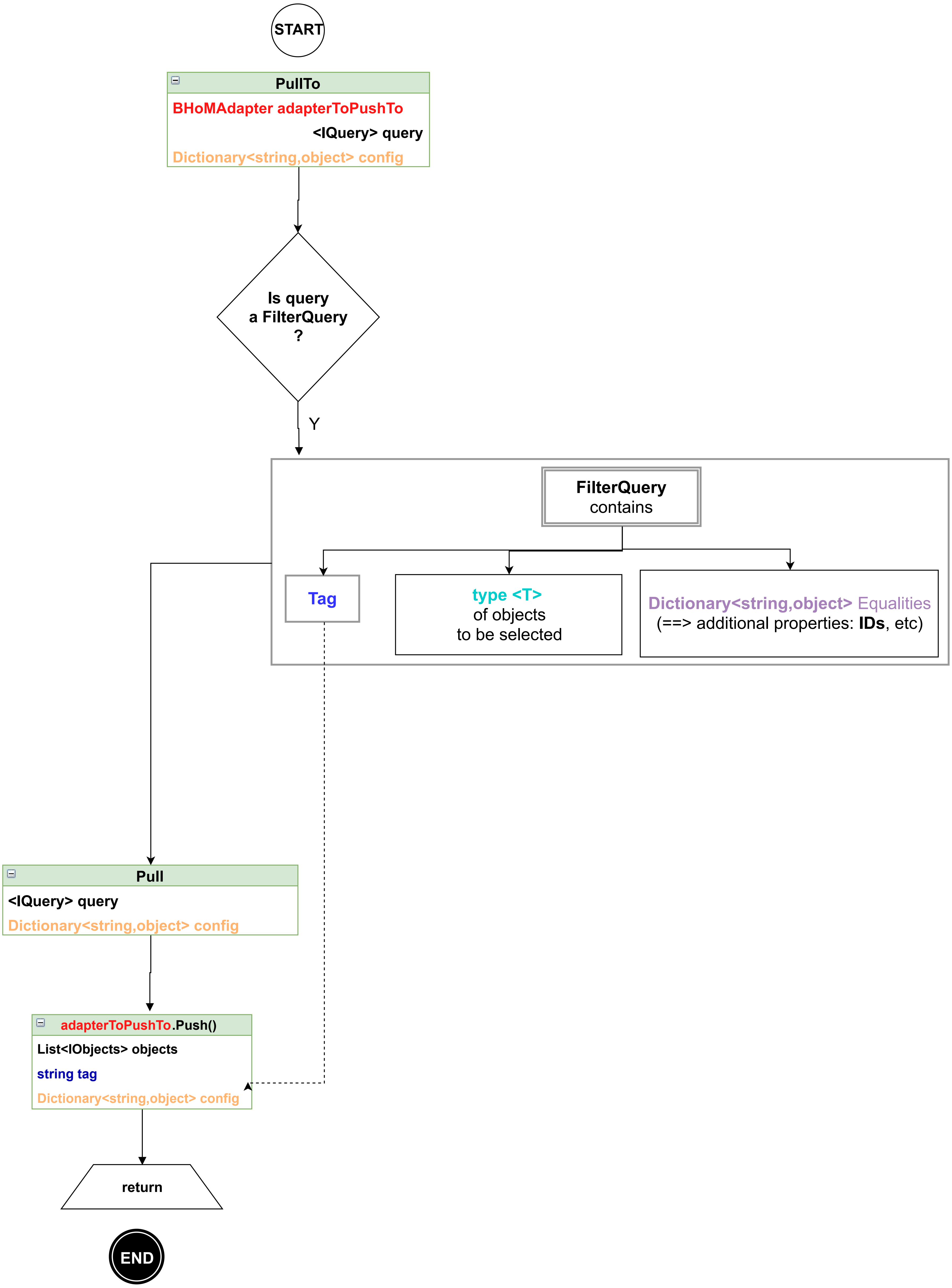
Main method performing the Read of an existing model.
Can be overridden in concrete adapter implementations.


START
Pull
<IQuery> query
Dictionary<string,object> config
Is query a FilterQuery ?
Y
FilterQuery contains
Tag
type <T> of objects to be selected
Dictionary<string,object> Equalities (==> additional properties: IDs, etc)
Get IDs from FilterQuery
Check type <T> of objects to be selected
IResult
ReadResults
<T> type of objects
IDs
(Retrieve objects Existing in the model of same type of <T> and filter them out with specified IDs)
return
List<IResult> pulled results
END
IBHoMObject
Read
<T> type of objects
IDs
(Retrieve objects Existing in the model of same type of <T> and filter them out with specified IDs)
return
List<IBHoMObject> pulledObjects
END

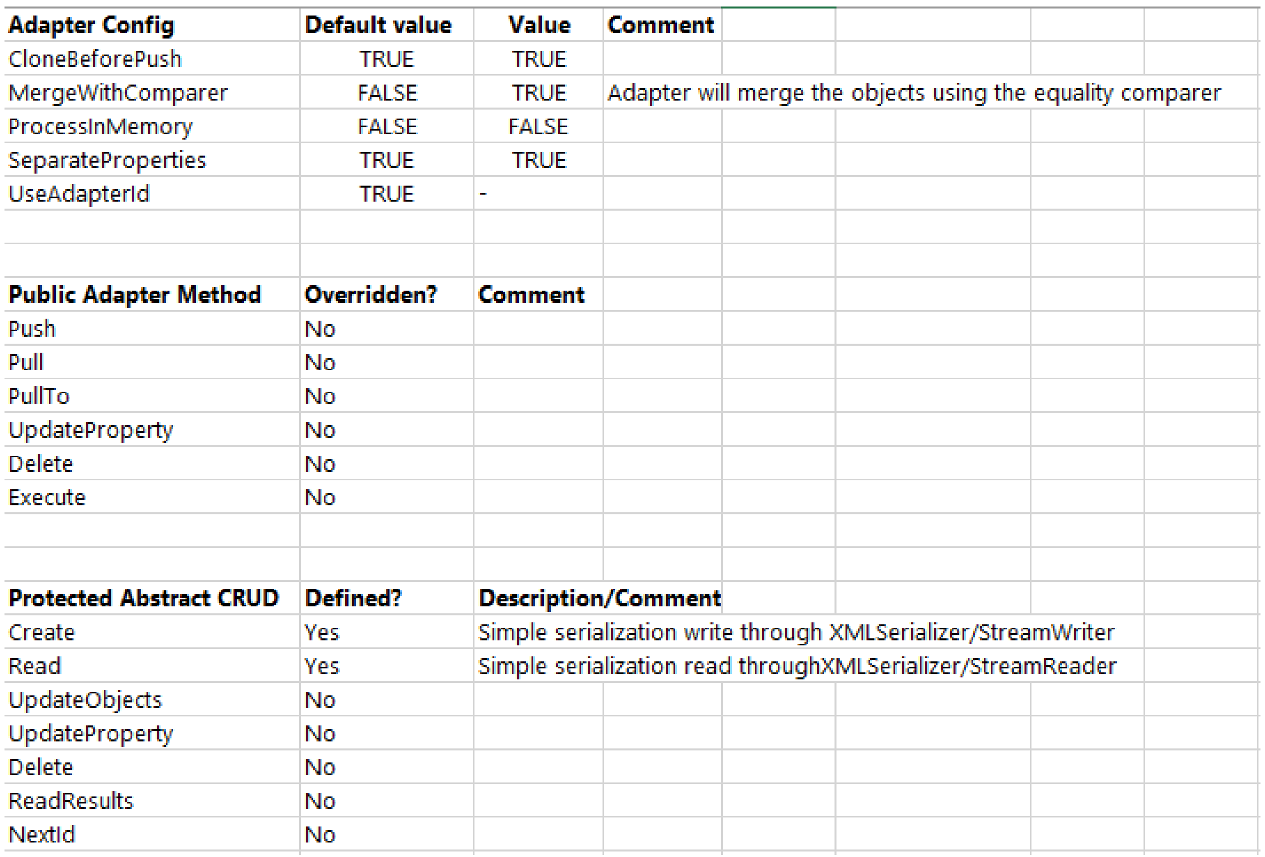
```
// Collect the objects from this
```

PULLTO

Main method performing the Read of an existing model.
Can be overridden in concrete adapter implementations.


START
PullTo
BHoMAdapter adapterToPushTo
<IQuery> query
Dictionary<string,object> config
Is query a FilterQuery ?
Y
FilterQuery contains
Tag
type <T> of objects to be selected
Dictionary<string,object> Equalities (==> additional properties: IDs, etc)
Pull
<IQuery> query
Dictionary<string,object> config
adapterToPushTo.Push()
List<IObjects> objects
string tag
Dictionary<string,object> config
return
END

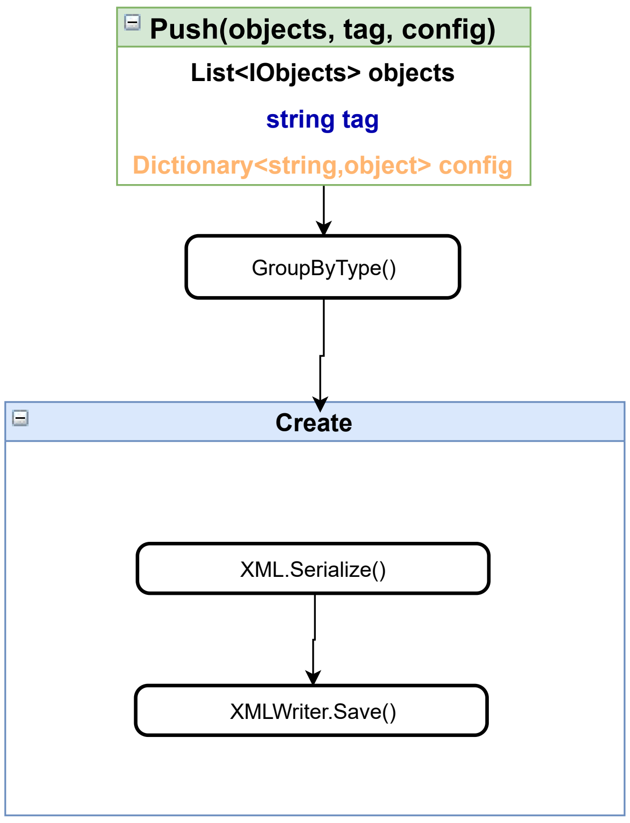
XMLAdapter

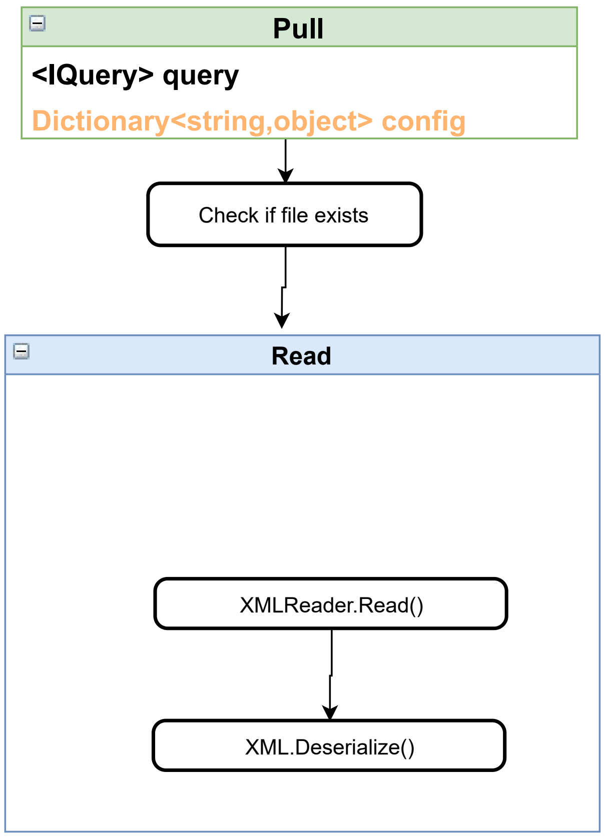


Adapter Config	Default value	Value	Comment
CloneBeforePush	TRUE	TRUE	
MergeWithComparer	FALSE	TRUE	Adapter will merge the objects using the equality comparer
ProcessInMemory	FALSE	FALSE	
SeparateProperties	TRUE	TRUE	
UseAdapterId	TRUE	-	

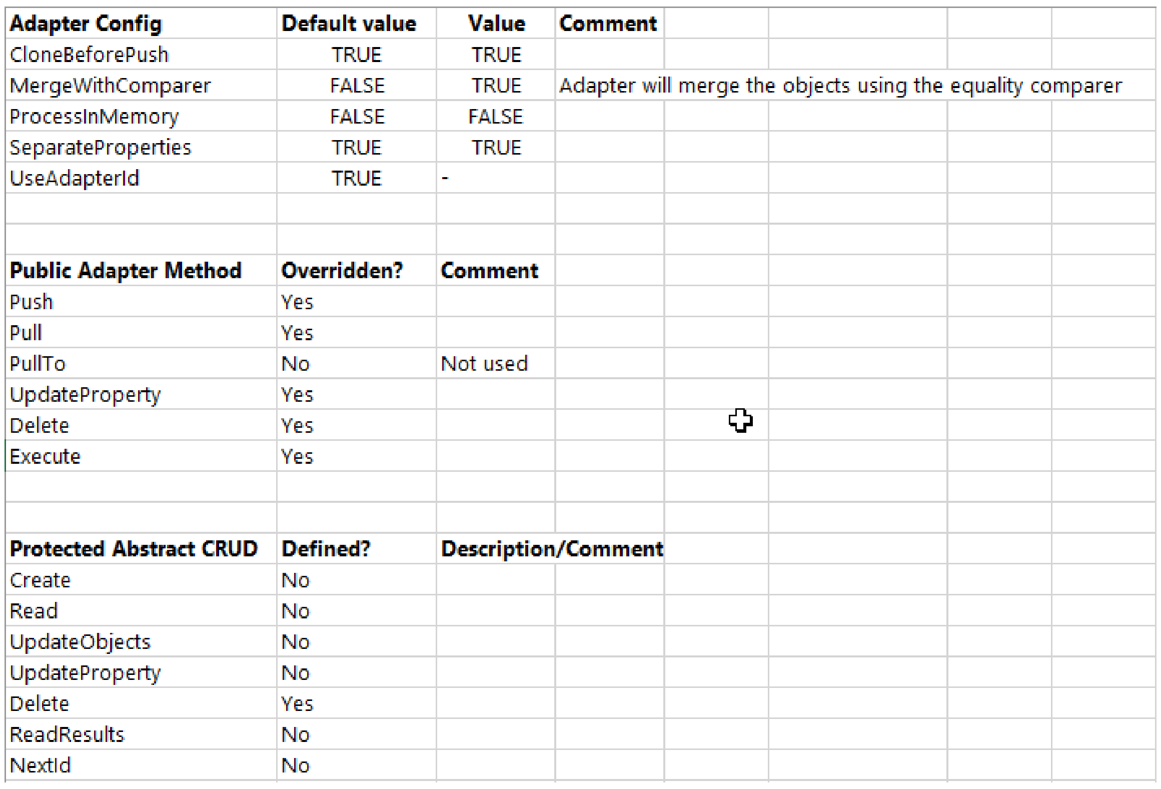
Public Adapter Method	Overridden?	Comment
Push	No	
Pull	No	
PullTo	No	
UpdateProperty	No	
Delete	No	
Execute	No	

Protected Abstract CRUD	Defined?	Description/Comment
Create	Yes	Simple serialization write through XMLSerializer/StreamWriter
Read	Yes	Simple serialization read throughXMLSerializer/StreamReader
UpdateObjects	No	
UpdateProperty	No	
Delete	No	
ReadResults	No	
NextId	No	





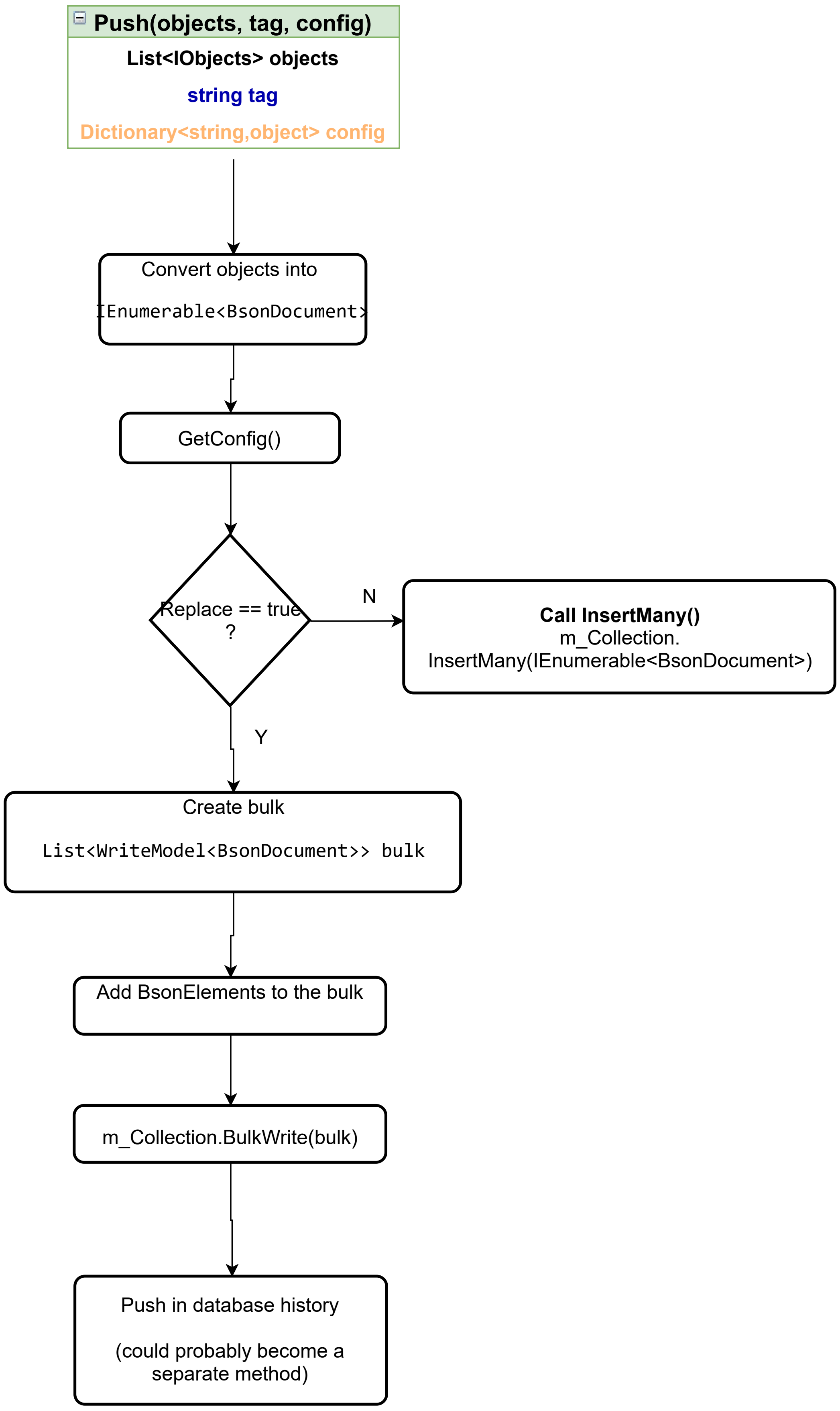
Mongo Adapter



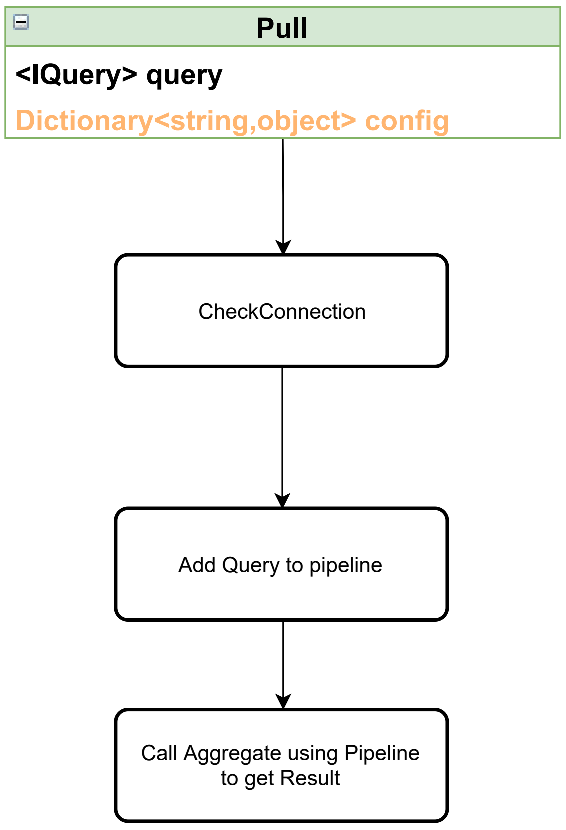
Adapter Config	Default value	Value	Comment
CloneBeforePush	TRUE	TRUE	
MergeWithComparer	FALSE	TRUE	Adapter will merge the objects using the equality comparer
ProcessInMemory	FALSE	FALSE	
SeparateProperties	TRUE	TRUE	
UseAdapterId	TRUE	-	

Public Adapter Method	Overridden?	Comment
Push	Yes	
Pull	Yes	
PullTo	No	Not used
UpdateProperty	Yes	
Delete	Yes	
Execute	Yes	

Protected Abstract CRUD	Defined?	Description/Comment
Create	No	
Read	No	
UpdateObjects	No	
UpdateProperty	No	
Delete	Yes	
ReadResults	No	
NextId	No	

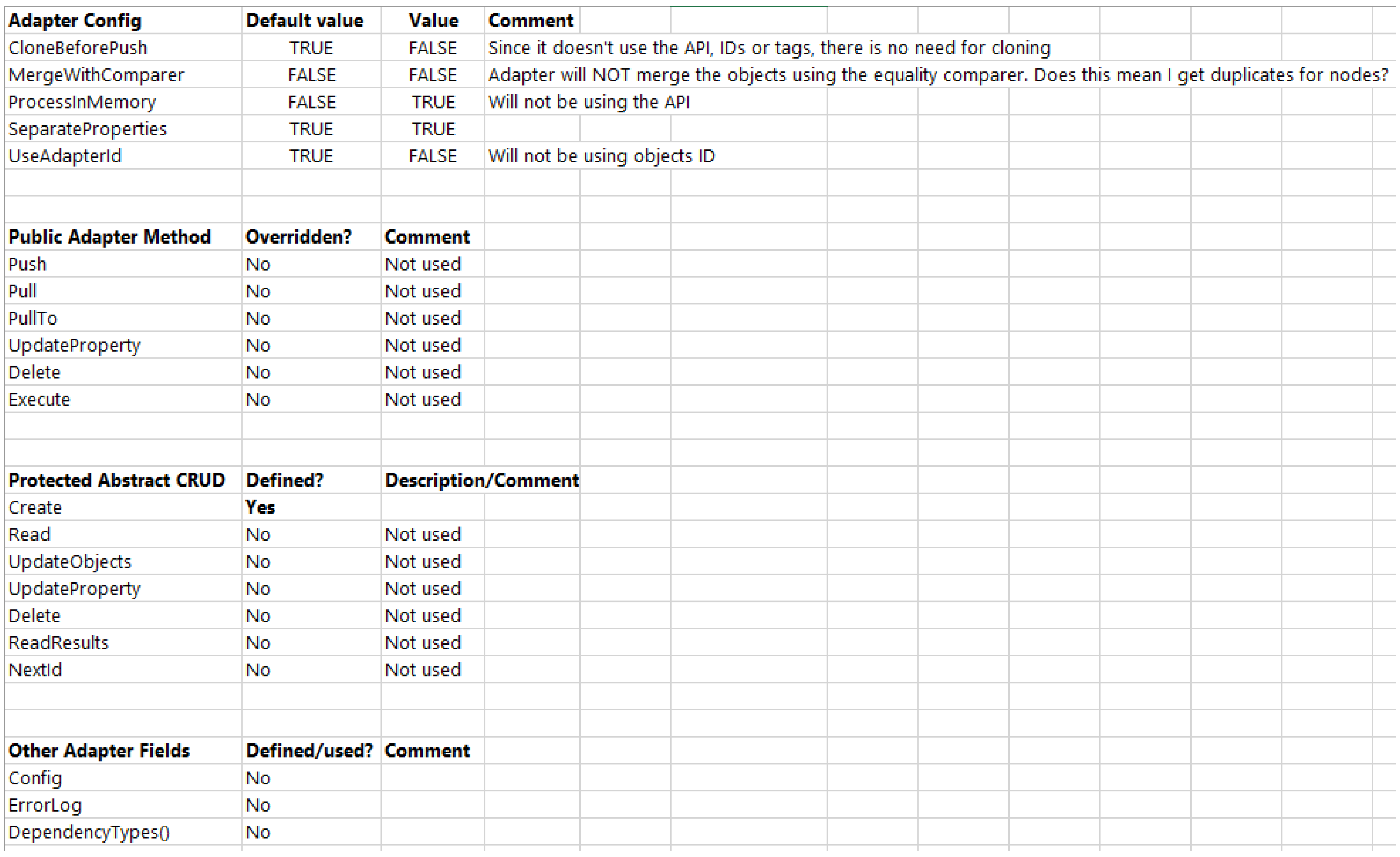


Push(objects, tag, config)
List<IObjects> objects
string tag
Dictionary<string,object> config
Convert objects into
IEnumerable<BsonDocument>
GetConfig()
Replace == true
?
N
Call InsertMany()
m_Collection.
InsertMany(IEnumerable<BsonDocument>)
Y
Create bulk
List<WriteModel<BsonDocument>> bulk
Add BsonElements to the bulk
m_Collection.BulkWrite(bulk)
Push in database history
(could probably become a separate method)

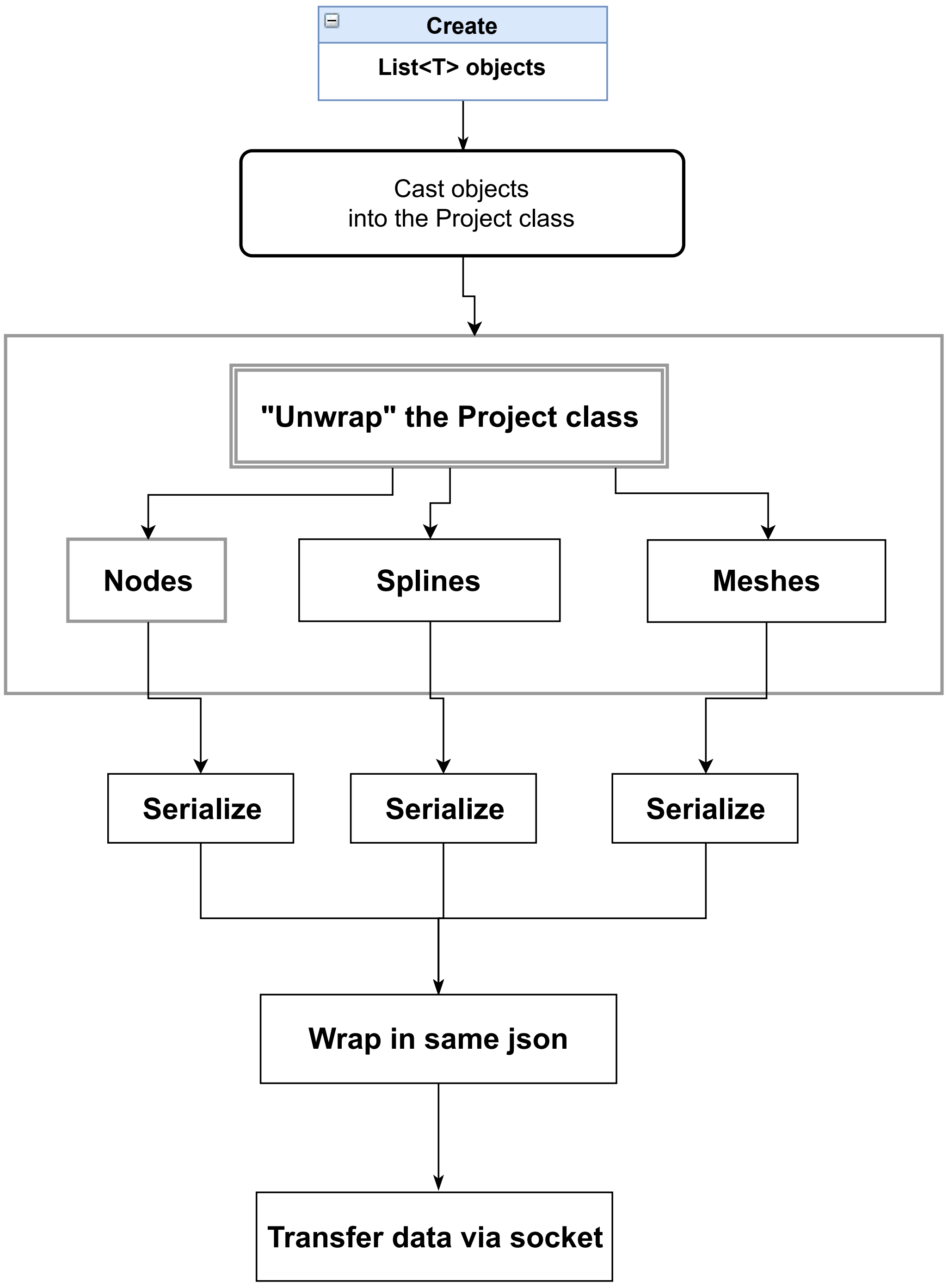


Pull
<IQuery> query
Dictionary<string,object> config
CheckConnection
Add Query to pipeline
Call Aggregate using Pipeline to get Result

Unreal_Adapter

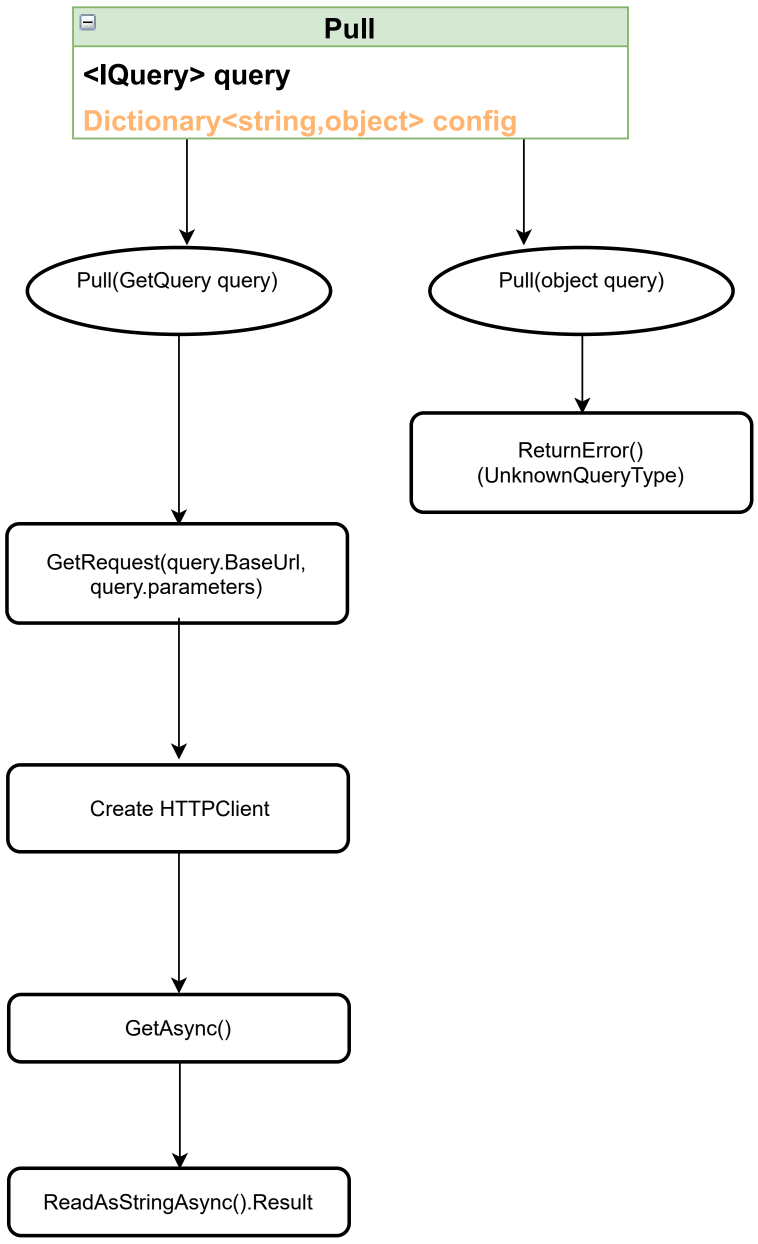


Adapter Config	**Default value**	**Value**	**Comment**
CloneBeforePush	TRUE	FALSE	Since it doesn't use the API, IDs or tags, there is no need for cloning
MergeWithComparer	FALSE	FALSE	Adapter will NOT merge the objects using the equality comparer. Does this mean I get duplicates for nodes?
ProcessInMemory	FALSE	TRUE	Will not be using the API
SeparateProperties	TRUE	TRUE	
UseAdapterId	TRUE	FALSE	Will not be using objects ID
Public Adapter Method	**Overridden?**	**Comment**	
Push	No	Not used	
Pull	No	Not used	
PullTo	No	Not used	
UpdateProperty	No	Not used	
Delete	No	Not used	
Execute	No	Not used	
Protected Abstract CRUD	**Defined?**	**Description/Comment**	
Create	**Yes**		
Read	No	Not used	
UpdateObjects	No	Not used	
UpdateProperty	No	Not used	
Delete	No	Not used	
ReadResults	No	Not used	
NextId	No	Not used	
Other Adapter Fields	**Defined/used?**	**Comment**	
Config	No		
ErrorLog	No		
DependencyTypes()	No		



HTTP_Toolkit

- Push not implemented (POST request not implemented)
- Overrides Pull: implements a runtime Multiple Dispatch depending on query type ("overload" of different Pull methods)
- No Create or Read implemented

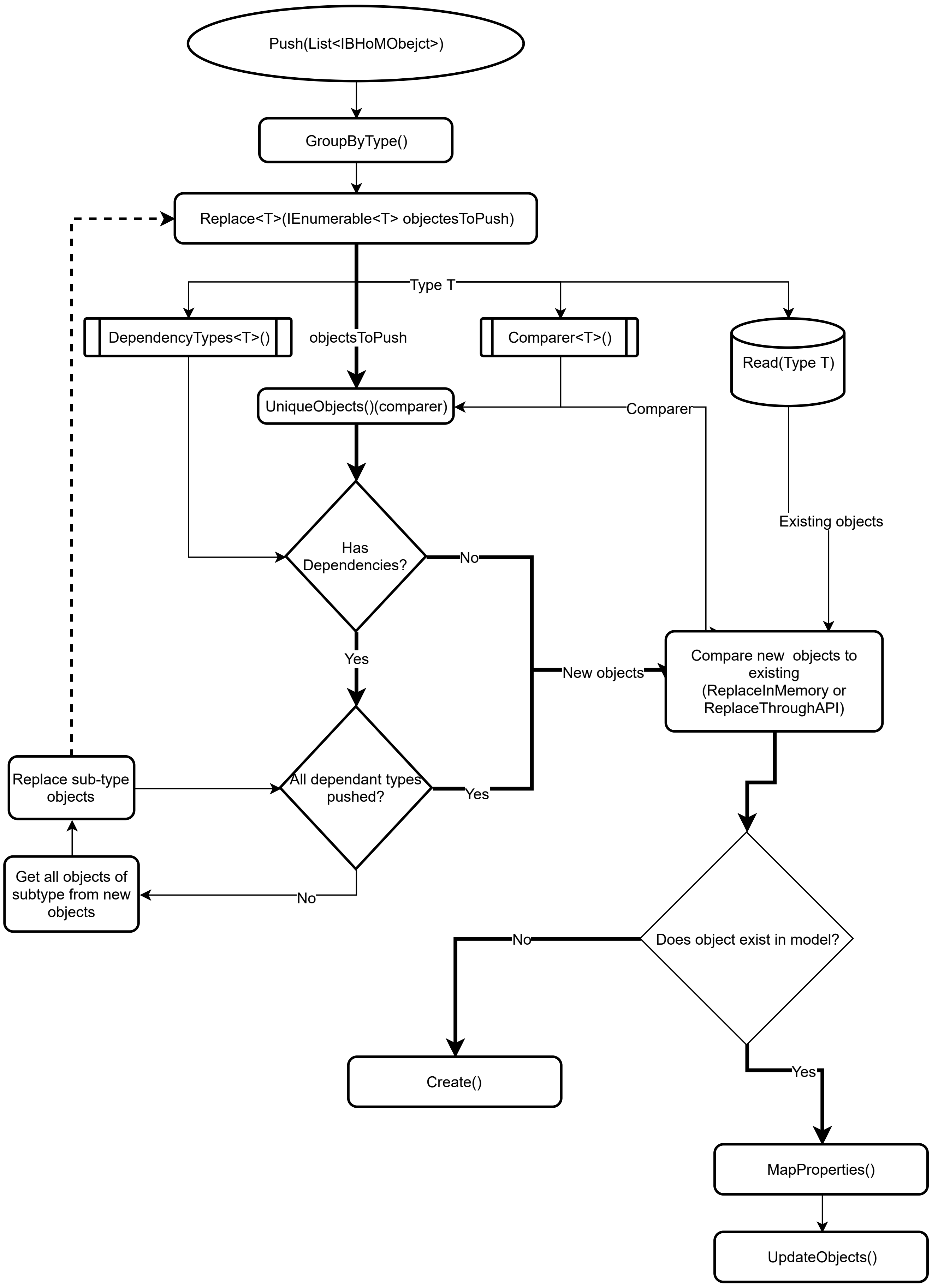


Pull
<IQuery> query
Dictionary<string,object> config
Pull(GetQuery query)
Pull(object query)
ReturnError() (UnknownQueryType)
GetRequest(query.BaseUrl, query.parameters)
Create HTTPClient
GetAsync()
ReadAsStringAsync().Result

Structural Adapters in general (GSA, Robot, ETABS)

- Structural adapters do not override the Push/Pull methods
- GSA/Robot/ETABS/Tekla

Push



Push(List<IBHoMObejct>)
GroupByType()
Replace<T>(IEnumerable<T> objectesToPush)
Type T
DependencyTypes<T>()
objectsToPush
Comparer<T>()
Read(Type T)
UniqueObjects()(comparer)
Comparer
Existing objects
Has Dependencies?
No
Yes
New objects
Compare new objects to existing (ReplaceInMemory or ReplaceThroughAPI)
Replace sub-type objects
All dependant types pushed?
Yes
Get all objects of subtype from new objects
No
Does object exist in model?
No
Create()
Yes
MapProperties()
UpdateObjects()

```
propertypropertyObjects.AddRange(objects.SelectMany(x=>get

modify what's inside the Addrange to match whatever it's ne
Prop(x)));

eeded by the specific type
```

Questioning the ID reapplyinment?
SeparateProperties & DependancyTypes can be toggled to fully not use the IDs already

You create the dependencies before the parents
--> this is why you can rely on having correct IDs for objects when coding

Selective DeepCloning: keep track of the types that need to be cloned and of those that do not

How to have a deepcloning changes the diagrams

Then focus on the deepcloning as a complete isolated problem
The only relation with adapters is that's i'ts a selectiveCloning -> how to do a selection