

```
//
// FILENAME: Inspectable.h
// SUMMARY: Defines a protocol for any inspectable object in eText.
// SUPERCLASS:None
// PROTOCOLS: <Inspectable, InspectableTarget>
// INTERFACE: None
// AUTHOR: Rohit Khare
// COPYRIGHT: ©1993,94 California Institute of Technology, eText Project
//
// Description
//
// - (const NXAtom *) types;
// - inspectorForType:(const char *) type;
// - resignInspector: (View *) oldInspector ofType: (const char *) type;
// - activateInspector:(View *) newInspector ofType:(const char *) type;
// - (char *) inspectorTitle;
//
// This is a protocol outlining the responsibilities of any object
// that wishes to attach itself to the Inspector facility of most
// ETF-applications. This protocol defines a standard interface for objects
// to advertise actual inspectors, and for the system to notify the
// inspectors' owners.
//
// Over time, the system has evolved into a strict MVC model, and the
// <Inspectable> object is often a shared interface controller that can
// dynamically "contain" the data of many clients (the */*UI pairs in eText).
// So, we have a concrete problem with persistent selections, since flipping
// back and forth between two documents with selected objects of the same
// class causes problems -- the <Inspectable> object is the same, the content
// is not. So, we developed the <InspectableTarget> extension.
//
// [inspector inspect:foo] will now stop to ask: does foo have an
// inspectableDelegate method? If so, store foo as the current target,
// and use the return value ([[fooUI new] setFoo:foo]) as the <Inspectable>.
//
// NS: The medium for an inspector is a View. For ease of use with Interface
// Builder, we suggest the programmer create a Panel or Box of the needed
// size, and then pass in the contentView of these objects.
//
// HISTORY
// 10/30/94: Created <InspectorTarget> protocol.
// 09/27/94: Revamped for eText5
// 08/18/93: Created. Derived almost wholly from Version 1.0
// Version 1.0 tracking notes:
// 06/29/93: Added -doc for the purpose of Inspector's -touch
// 06/27/93: Created (Rohit Khare)
//
// Imported Interfaces
```

```

//
    #import <appkit/View.h>
    #import <objc/hashtable.h>

// NS: These are sized in relation to the Workspace Manager's inspector panels.
#define INSPECTOR_W    270.0
#define INSPECTOR_H    378.0

@protocol Inspectable

- (const char *) inspectorTitle;
// General: The panel will put "inspectorTitle Inspector" in the window bar.
// Note: the return value should be less than 240 characters.

- (const NXAtom *) types;
// General: Inspectable objects can have several different inspectors.
// A common set might be "Info, Controls, Contents" Controls would be UI
// interactors (Play, Pause, etc.) and Contents might include filenames,
// formats, etc. The first type in the list is the "primary" type, which is the
// one a user is presented with first.
//
// NS: This list is loaded into a popup list, and the first item
// on the list is the default view.

- inspectorForType: (const char *) type;
// General: For each type of inspector, the system will automatically request
// the needed panel on demand. The returned object is an inspector.
//
// NS: The offered view will be forcibly resized to the panel's full size.

- resignInspector: (View *) oldInspector ofType: (const char *) type;
- activateInspector: (View *) newInspector ofType: (const char *) type;
// General: The object which is being inspected also will be notified of status
// changes. The inspected object is the implicit delegate of the inspector
// controller. The arguments are the view and the tag given by the programmer
// in methods above.

@end

@protocol InspectableTarget

- (id <Inspectable>) inspectableDelegate;
// General: This is a "standard" proxy indirection mechanism for separating
// the inspected object from its inspectable interface (foo from fooUI)

@end

```