HTML5 Zero-footprint Viewer for DICOM and PACS

LEADTOOLS®
THE WORLD LEADER IN IMAGING SDKs

LEAD TECHNOLOGIES INCORPORATED

1

Introduction

The demand for zero-footprint applications is growing rapidly, especially in the healthcare industry with the increasing popularity and usage of tablets and mobile devices by healthcare providers. There are many ways to display DICOM images and communicate with a PACS over the web or intranet in a zero-footprint manner, but not all applications are created equal.

Any zero-footprint application must do a significant portion of work at the server, but this typically causes a tradeoff in features and performance for which developers must find the proper balance. When simplifying deployment and platform compatibility through server-side processing, the performance and user-friendliness found in a desktop or rich client application is typically lost or degraded. Additionally, DICOM images pose a unique challenge because of the 16-bit grayscale image data typically found within them. These images must be window leveled in order to be properly displayed on standard screens that are only capable of interpreting 8 bits of grayscale data. The most common solution is to use AJAX to send the window level values to the server and then asynchronously get the converted image back. Unfortunately this method is slow and unresponsive, especially with larger images.

LEADTOOLS HTML5 Zero-footprint DICOM Viewer: a Comprehensive Solution

LEADTOOLS overcomes these problems with an advanced JavaScript library that is able to window level and process 16-bit grayscale images on an HTML5 canvas. This speedy, industry-leading technology runs completely in the client's browser and requires no plug-ins while achieving a uniquely responsive user experience. LEADTOOLS also provides a sizable collection of client-side tools including image stack, zoom, pan, magnifying glass, annotations and markup, reference lines, multi-frame layouts, cine and more.

The LEADTOOLS HTML5 Zero-footprint DICOM viewer is a fully functional AngularJS web application that integrates directly with any PACS to stream DICOM images to the client. The source code is provided so that developers can easily make modifications, customizations and branding changes to the application as they see fit.

2

Using the RESTful Web Service to Query and Retrieve DICOM Images

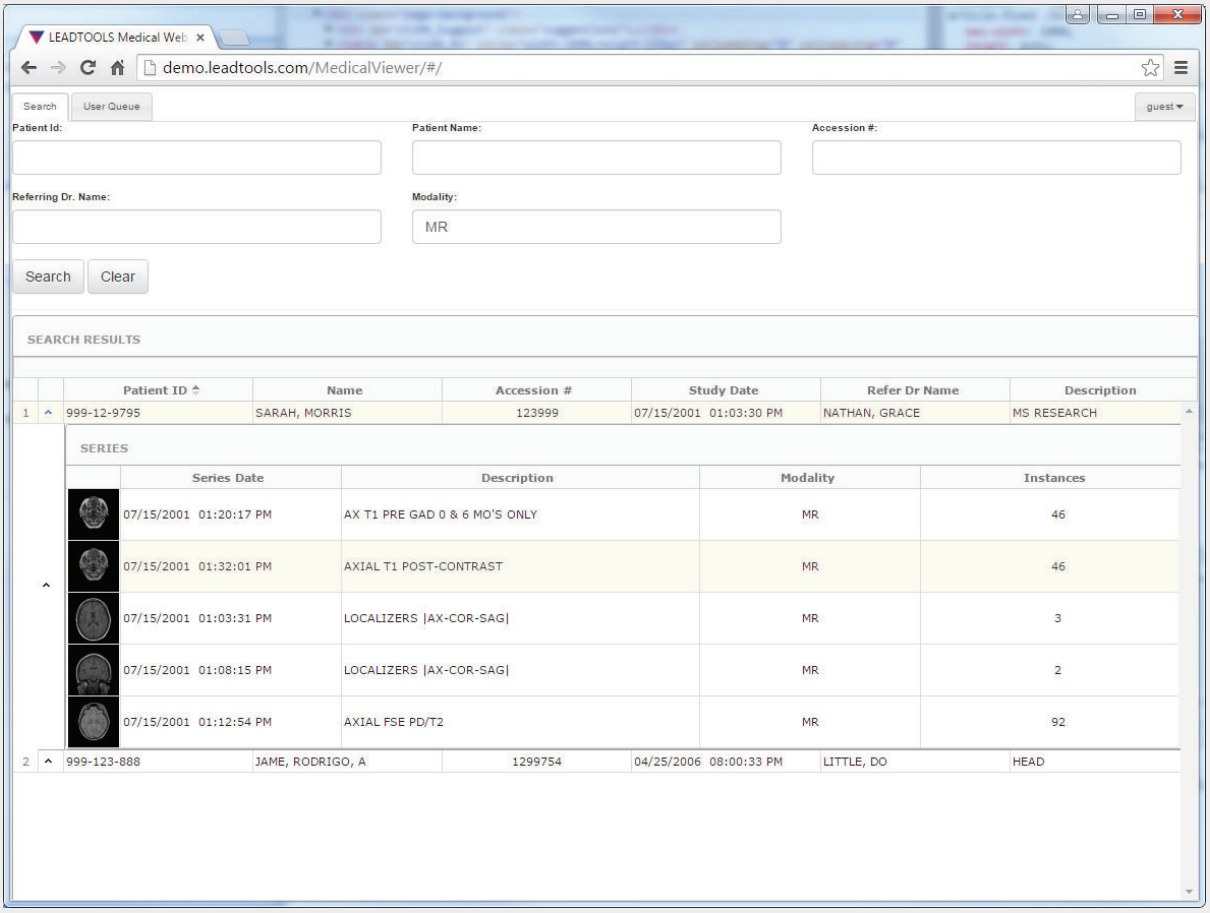
The server component uses a RESTful Web Service to interface with a local archive or any remote PACS to which you have access. This service handles all of the PACS communication (i.e. C-Find, C-Move, C-Store, etc.) and interfaces with the viewer control using JSON.

After searching the archive and selecting a patient, study and series, the images will start streaming to the viewer. First, the server will send a JPEG compressed image to be displayed immediately, and in the background the server streams the window level data (more details to follow in the next section) and remaining image frames in the series stack.

With the MVC architecture afforded by AngularJS, querying the server is quite simple. Each `input` in the view is tied to a `QueryOptions` model. Then, the form's search button is attached to the `doSearch` controller function shown in the snippet below, which interacts with the database or remote PACS. Following that, the application processes incoming server responses and populates the studies and series lists.

```
$scope.doSearch = function () {
   // Get query options from inputs
   var queryOptions = angular.copy($scope.queryOptions);

   // Empty the currently displayed search results
   $scope.studies.length = 0;

   // Query the database directly or a remote PACS
   switch ($scope.querySource.name) {
      case 'database':
         queryArchiveService.FindStudies(queryOptions,
               maxStudyResults).then(function (result) {
            eventService.publish("Search/Study/Success", result.data);
            $scope.studies = result.data; // Populate the new results
         }, function (error) {
            eventService.publish("Search/Study/Failure", { error: error });
         });
         break;
      case 'pacs':
         queryPacsService.FindStudies($scope.querySource.pacs,
               queryPacsService.clientAETitle,
               queryOptions).then(function (result) {
            eventService.publish("Search/Study/Success", result.data);
            $scope.studies = result.data; // Populate the new results
         }, function (error) {
            eventService.publish("Search/Study/Failure", { error: error });
         });
         break;
   }
};
```

3



Client-side DICOM Image Window Leveling

Since most DICOM Data Sets contain 16-bit grayscale image data and monitors only display 8 bits, window leveling is equally as important as the initial image display. Why is client-side window leveling so important? Without it, the parameters must be sent to the server, which does the window leveling, converts it to an 8-bit image and returns it to the client. As images get larger, and healthcare professionals use more restrictive hardware such as mobile phones and tablets, the performance will degrade rapidly as large amounts of data go back and forth between server and client.

LEADTOOLS alleviates this performance degradation by streaming the original DICOM image data using lossless compression and storing it in the client's cache. This way, the HTML5 viewer's window level interactive mode can interpret the mouse or touch input and resample the image colors on the client side, resulting in the same speed and responsiveness you would get from a desktop application.

When a series is selected for viewing, the controller will loop through all of the instances and frames and add them to the cell. It will set the DICOM data for the first cell and stream the remaining images as other requests are made.

```
DicomLoaderService.prototype.LoadImages = function (seriesInstanceUID, xmlData) {
   var instances = this._seriesManagerService.get_instances(seriesInstanceUID);
   var cell = this._seriesManagerService.get_seriesCell(seriesInstanceUID);

   for (var instanceIndex = 0; instanceIndex < instances.length; instanceIndex++) {
      var instance = instances[instanceIndex];

      for (var frameIndex = 0; frameIndex < instance.NumberOfFrames; frameIndex++){
         // Set the DICOM data if this is the first instance, otherwise null
         cellFrame.DicomData = (instanceIndex == 0) ? xmlData : null;
         cellFrame.Instance = instance;
         cellFrame.enableDraw = true;

         cellFrame.add_imageDrawn($.proxy(this.OnImageDrawn, this));
      }
   }

   this._eventService.publish(
      EventNames.LoadingSeriesFrames, { seriesInstanceUID: seriesInstanceUID });
};
```

While all this is going on, the user may notice the viewer's window level button is initially disabled when the image is first displayed or scrolled into view. Upon initial viewing, `loadFrameData` triggers an asynchronous request to the server and the raw DICOM image data starts streaming to the client and after a second or two, the button is enabled and a label is displayed showing the current window level values.

```
DicomLoader.prototype.loadFrameData = function (cellFrame, dataSize, url) {
   var deferred = this._qService.defer();

   var dataReady = function (e) {
      cellFrame.remove_imageDataReady(dataReady);
      cellFrame.remove_imageDataError(dataError);
      deferred.resolve(e);
      this._eventService.publish(
         EventNames.ImageDataReady, { cellFrame: cellFrame });
   };

   cellFrame.add_imageDataReady(dataReady);
   cellFrame.setPNGDataSrc(url, dataSize.width, dataSize.height);
   return deferred.promise;
};
```

5

6

HTML5 Annotations for DICOM Images

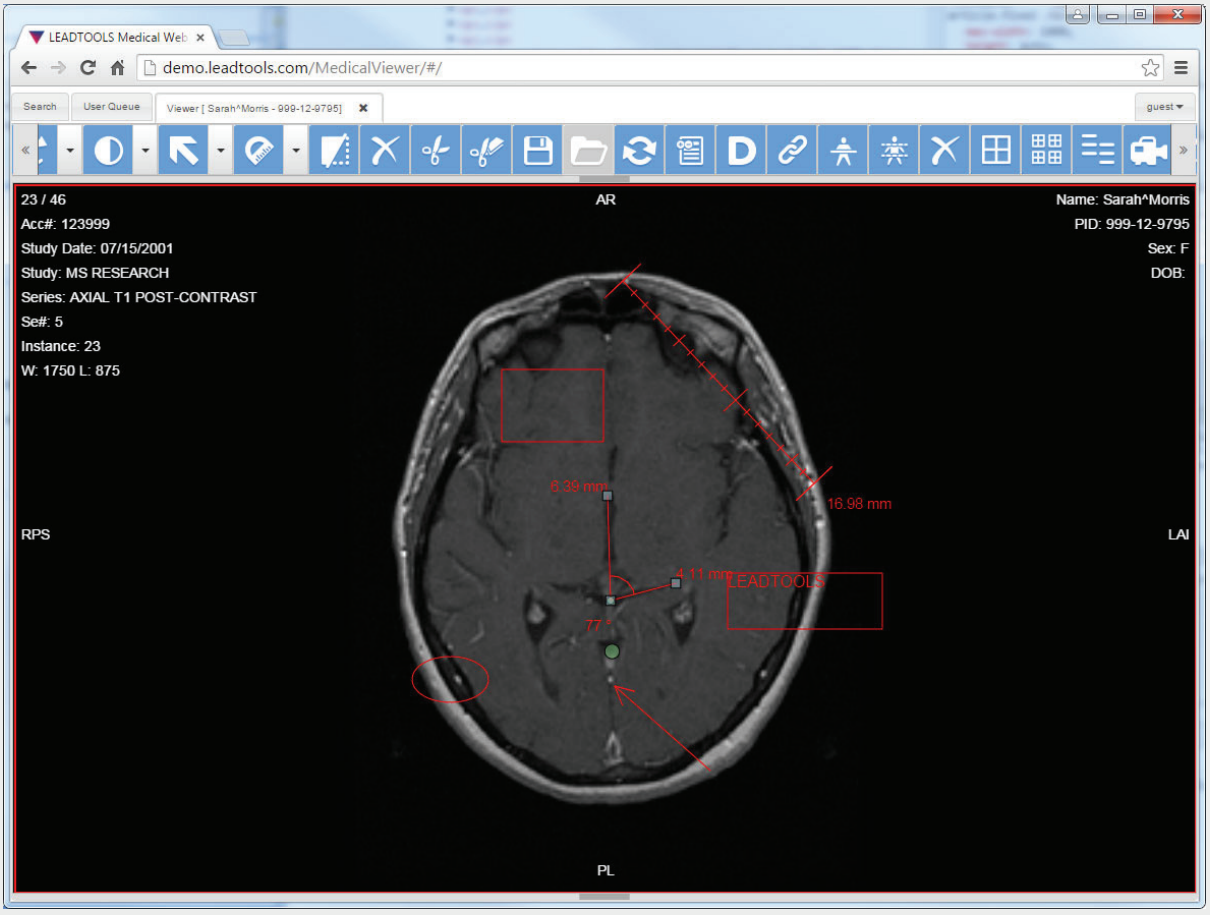
Once a DICOM series has been selected and the images begin streaming to the viewer, the annotations are initialized for use. The `AnnAutomationManager` object is created and attached to the viewer. The annotations are given their own HTML5 canvas which is overlaid on top of the viewer. This allows the annotations to be drawn in a separate layer from the image and increases efficiency and reduces the possibility for corruption of the canvas being displayed underneath.

The great thing about the `AnnAutomationManager` is that it does everything for you. All the events are handled internally so mouse and touch events are correctly interpreted to draw, modify, transform and scale the annotations anytime the user interacts with the canvas or annotation objects. Additionally, it will rescale and translate the annotations accordingly whenever the viewer's display properties such as zooming and scrolling are altered so that the annotations stay in the same logical position on the image.

To use the annotations, all one must do next is select the object you wish to draw, or use the Select tool to modify an existing annotation. In the demo application, LEADTOOLS includes several buttons that enable the desired annotation tool. You can enable or disable as many as you wish, but LEADTOOLS ships the demo with the most popular annotations used in the healthcare industry (Arrow, Rectangle, Ellipse, Text, Highlight, Ruler, Poly Ruler and Protractor). The snippet below shows the Angular click commands of several buttons to give an idea of how little is needed for so many features.

```
commangular.command('OnAnnotationSelect', [
   'toolbarService', 'tabService', 'buttonId',
   function (toolbarService, tabService, buttonId) {
      return {
         execute: function () {
            setAnnTool(toolbarService, tabService,
               buttonId, MedicalViewerAction.AnnSelect);
         }
      };
   }]);

commangular.command('OnAnnotationArrow', [
   'toolbarService', 'tabService', 'buttonId',
   function (toolbarService, tabService, buttonId) {
      return {
         execute: function () {
            setAnnTool(toolbarService, tabService,
               buttonId, MedicalViewerAction.AnnPointer);
         }
      };
   }]);
```

```
commangular.command('OnAnnotationText', [
   'toolbarService', 'tabService', 'buttonId',
   function (toolbarService, tabService, buttonId) {
      return {
         execute: function () {
            setAnnTool(toolbarService, tabService,
               buttonId, MedicalViewerAction.AnnText);
         }
      };
   }]);
```



Loading and Saving Annotations Using the Web Service

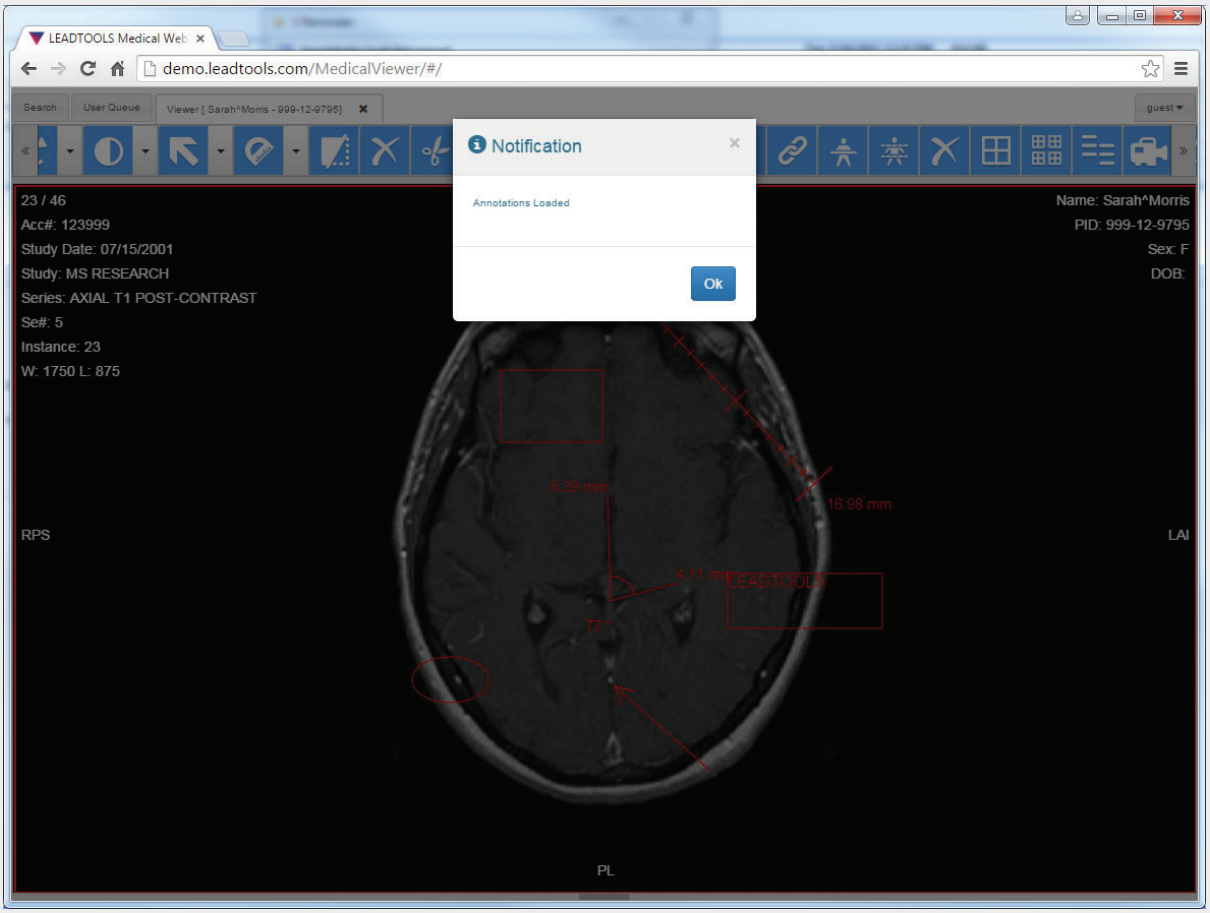
The ability to load and save annotations is crucial to the workflow of medical applications. First and foremost, they help describe, point out, or make note of something in the image. The most important piece of information is still the image itself so annotations should have a simple method of being hidden and brought back. DICOM viewing applications are also collaborative. Radiologists, nurses, doctors and patients alike can look at the images and often need to get second opinions, making the ability to pass notes and annotations back and forth very handy. Finally, this is a web application so the users of the application will need to see the image and annotations on any computer, mobile device or tablet from any location.

LEADTOOLS uses a RESTful web service to load and save the annotations. As shown below, the first step is to get a description (e.g. “Dr. Brown’s Notes”, “John please review!”, etc.) and the image frame on which the annotations are drawn (SOP Instance UID). These two pieces of information are passed to the `ObjectStoreService`, which takes care of the rest by saving the annotation data to the server’s database.

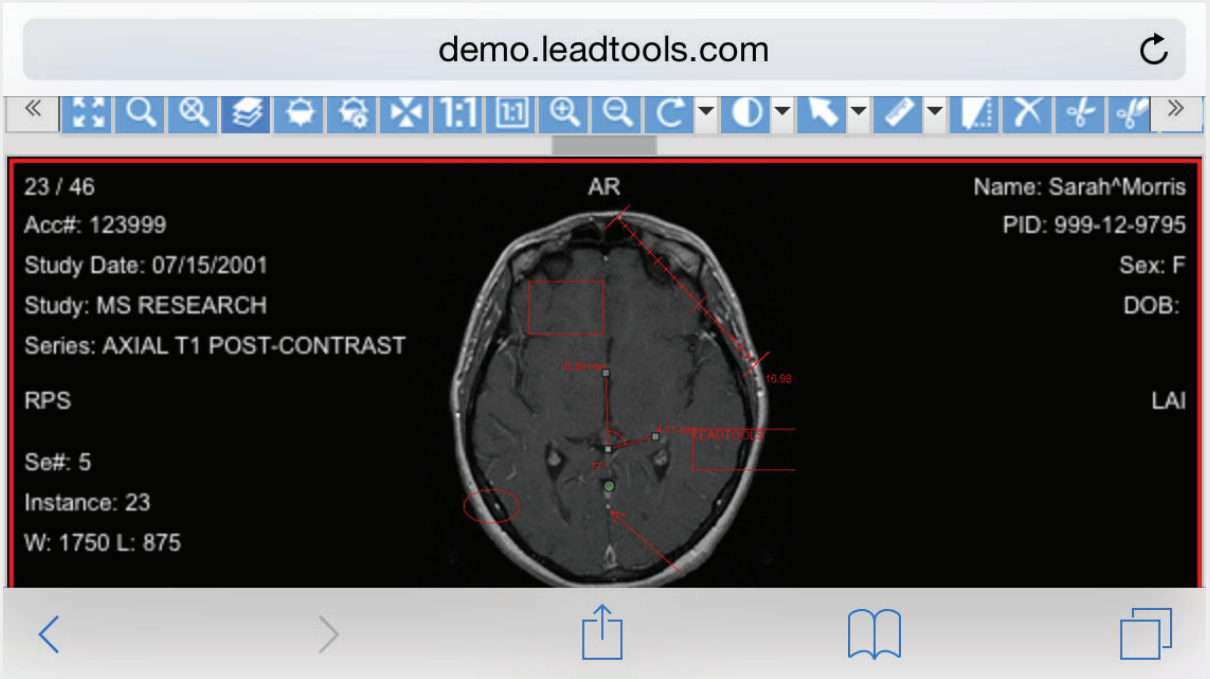
```
commangular.command('OnSaveAnnotations', [
   'seriesManagerService', 'toolbarService', 'objectStoreService',
   'authenticationService', '$modal', '$translate', 'dialogs', 'tabService',
   'optionsService',
   function (seriesManagerService, toolbarService, objectStoreService,
         authenticationService, $modal, $translate, dialogs, tabService,
         optionsService) {
      return {
         execute: function () {
            var tab = tabService.get_allTabs()[tabService.activeTab];

            if (toolbarService.isEnabled("SaveAnn" + tab.id)) {
               var cell = seriesManagerService.get_activeSeriesCell();

               if (cell) {
                  // Save annotations to this series
                  var seriesInstanceUID = cell.get_seriesInstanceUID();
                  var annotationsData =
                     seriesManagerService.get_annotationsData(seriesInstanceUID);

                  if (annotationsData.length > 0) {
                     // Get description from user
                     var modalInstance = $modal.open({
                        templateUrl: 'views/dialogs/AnnotationsSave.html',
                        controller: Controllers.AnnotationsSaveController,
                        backdrop: 'static'
                     });

                     modalInstance.result.then(function (description) {
                        objectStoreService.StoreAnnotations(seriesInstanceUID,
                              annotationsData, description).then(function (result){
                           seriesManagerService.add_annotationID(
                              seriesInstanceUID, result.data);
                           dialogs.notify(notifyTitle, annotationsSaved);
                        });
                     });
                  }
               }
            }
         }
      };
   }]);
```

When an image frame is loaded, the application does a quick permissions check and then retrieves an array of previously saved annotation files associated with the image. If the image has annotations, then the load button is enabled. After the user selects an annotation file, the following code will get the annotation data from the server and add each annotation to the canvas.

```
commangular.command('OnLoadAnnotations', [
   'seriesManagerService', 'toolbarService', '$modal', 'eventService',
   'objectRetrieveService', '$translate', 'dialogs', 'tabService',
   function (seriesManagerService, toolbarService, $modal, eventService,
         objectRetrieveService, $translate, dialogs, tabService) {
      return {
         execute: function () {
            var tab = tabService.get_allTabs()[tabService.activeTab];

            if (toolbarService.isEnabled("LoadAnn" + tab.id)) {
               // Get annotations from this series
               var seriesInstanceUID = seriesManagerService.get_activeSeriesCell().
get_seriesInstanceUID();
               var annotations =
                  seriesManagerService.get_annotationIDs(seriesInstanceUID);

               objectRetrieveService.GetPresentationAnnotations(sopInstanceUID,
                     '').then(function (result) {
                  if (result.status == 200) {
                     if (result.data && result.data.length > 0) {
                        var xmlAnnotations = $.parseXML(result.data);

                        seriesManagerService.add_annotations(seriesInstanceUID,
                           xmlAnnotations);
                     }
                  }
               }, function (error) {
                  $translate('DIALOGS_ERROR').then(function (translation) {
                     dialogs.error(translation, error);
                  });
               });
            }
         }
      };
   }]);
```



As you can see in the next screenshot, the same image and annotations load perfectly from an iPhone.



11

Conclusion

Querying and retrieving DICOM images from a PACS and viewing them in a robust, cross-platform solution is just one of many real-world solutions you can tackle with LEADTOOLS. Its state-of-the-art HTML5 Zero-footprint Viewer for DICOM and PACS makes it possible to provide all the speed and responsiveness required to accurately window level, process and annotate DICOM images without sacrificing any of the features healthcare professionals need. LEADTOOLS offers an incredible value with its comprehensive family of toolkits for raster, document, medical and multimedia imaging. For more information on how LEAD Technologies can image-enable your application and boost your ROI, visit ***www.leadtools.com*** to download a free evaluation, or give us a call at +1-704-332-5532.

Sales: (704) 332-5532
sales@leadtools.com

Support: (704) 372-9681
support@leadtools.com

LEAD Technologies, Inc.
1927 South Tryon Street
Suite 200
Charlotte, NC 28203

About LEAD Technologies

With a rich history of nearly 25 years, LEAD has established itself as the world's leading provider of software development toolkits for document, medical, multimedia, raster and vector imaging. LEAD's flagship product, LEADTOOLS, holds the top position in every major country throughout the world and boasts a healthy, diverse customer base and strong list of corporate partners including some of the largest and most influential organizations from around the globe.

LEADTOOLS®
THE WORLD LEADER IN IMAGING SDKs

LEAD TECHNOLOGIES INCORPORATED