

# Appendix A

## Callback function API

This appendix lists all of the callbacks in the Arrange API.

### Data Callbacks

Integer GetFieldTextLen(arNoteID note, arFieldID field);	4
Integer GetFieldText( arNoteID note, arFieldID field, Integer bufLen, OUT void* buffer );	4
arDate GetFieldDate(arNoteID note, arFieldID field);	4
arFloat GetFieldNumber(arNoteID note, arFieldID field);	4
Integer GetFieldListLen(arNoteID note, arFieldID field);	4
arNoteID GetFieldListEntry( arNoteID note, arFieldID field, Integer index );	4
OWN arListID GetFieldList(arNoteID note, arFieldID field);	5
OWN PicHandle GetFieldPict(arNoteID note, arFieldID field);	5
OWN Handle GetFieldAlias(arNoteID note, arFieldID field);	5
void SetFieldText( arNoteID note, arFieldID field, const char* text, Integer textLen, arSetFieldFlags flags );	5
void SetFieldDate( arNoteID note, arFieldID field, arDate date, arSetFieldFlags flags );	5
void SetFieldNumber( arNoteID note, arFieldID field, arFloat number, arSetFieldFlags flags );	5
void SetFieldList( arNoteID note, arFieldID field, arListID list, arSetFieldFlags flags );	6
void AddListFieldEntry( arNoteID note, arFieldID field, Integer index, arNoteID childNote, arSetFieldFlags flags );	6
void DeleteListFieldEntry( arNoteID note, arFieldID field, arNoteID childNote, arSetFieldFlags flags );	6
void MoveListFieldEntry( arNoteID note, arFieldID field, arNoteID childNote, Integer index, arSetFieldFlags flags );	6
void SetFieldPict( arNoteID note, arFieldID field, PicHandle pict, arSetFieldFlags flags );	7
void SetFieldAlias( arNoteID note, arFieldID field, Handle alias, arSetFieldFlags flags );	7
void ReplaceFieldText( arNoteID note, arFieldID field, Integer repStart, Integer repEnd, const char* insText, arSetFieldFlags flags);	7
void GetFieldStyle( arNoteID note, arFieldID field, OUT arFieldStyle* fieldStyle, Integer bufLen, OUT arCharStyle* charStyles );	7
void SetFieldStyle( arNoteID note, arFieldID field, const arFieldStyle* fieldStyle, Integer charStyleCount, const arCharStyle* charStyles, arSetFieldFlags flags );	8
Note Calls	
arNoteID CreateNote(arTypeID type, Boolean doGather);	8
arNoteID DuplicateNote(arNoteID note, Boolean doGather);	8
void ConvertNote(arNoteID note, arTypeID type, Boolean doGather);	9
void DestroyNote(arNoteID note);	9
void AddField(arNoteID note, arFieldID field, arFieldID nextField);	9
void RemoveField(arNoteID note, arFieldID field);	9
void HideField(arNoteID note, arFieldID field);	9
void MoveField(arNoteID note, arFieldID field, arFieldID nextField);	10
Integer CountNoteFields(arNoteID note);	10

arFieldID GetNoteField(arNoteID note, Integer index);	10
Boolean NoteHasField( arNoteID note, arFieldID field, Boolean onlyIfVisible );	10
Boolean NoteExists(arNoteID note);	10
arTypeID GetNoteType(arNoteID note);	10
void GetNoteInfo(arNoteID note, IO arNoteInfo *info);	10
void SetNoteInfo(arNoteID note, const arNoteInfo *info);	10
<b>Selection Calls</b>	
arWindowID GetActiveWindow();	11
arSelType GetSelection( OUT arNoteID *note, OUT arFieldID *field, OUT Integer *selStart, OUT Integer *selEnd );	11
OWN arPathID GetSelPath();	11
void GetSelEntry( Integer index, OUT arNoteID *note, OUT arFieldID *field, OUT arNoteID *parentNote );	11
void SelectObject( arNoteID parent, arFieldID field, arNoteID note, Boolean doDeselect );	12
Boolean ObjectIsSelected(arNoteID parent, arFieldID field, arNoteID note);	12
Boolean FlushSelection(Boolean alsoDrop);	12
void ShowPath( arPathID path, Boolean selectFlag, Integer textSelStart, Integer textSelEnd );	13
Integer GetSelText( Integer bufLen, OUT void* buffer, OUT Integer *selStart, OUT Integer *selEnd );	13
void SetSelText( const char* text, Integer textLen, Integer selStart, Integer selEnd );	13
void ReplaceSelText( const char* text, const char* undoName );	13
<b>Search Calls</b>	
Boolean NextAppearance(IO arPathID path, arDirCode direction);	14
Boolean FindText( IO arPathID path, arDirCode direction, IO Integer *selStart, IO Integer *selEnd, const char* searchString, arSearchFlags flags );	14
OWN arListID FilterList( arListID list, arTypeID matchType, arFilterFlags flags, Short clauseCount, const arFilterClause* clauses );	14
OWN arListID FilterNotes( arTypeID type, arFilterFlags flags, Short clauseCount, const arFilterClause* clauses );	14
OWN arListID SortNotes( arListID list, Short clauseCount, const arSortClause* clauses );	15
<b>System Object Calls</b>	
arNoteID GetBuiltInObject(arBuiltInObject whichObject);	15
arNoteID LookupObjectName( arObjectClass objClass, const char* name, Boolean allowPartial );	15
<b>Topics, folders, and views</b>	
arTopicID CreateTopic( const char* name, arTopicID parent, arTopicID successor, Boolean makeView );	15
arTopicID CreateFolder(const char* name, arTopicID successor);	16
void UpdateView(arTopicID view);	16
void GetTopicInfo(arTopicID topic, IO arTopicInfo *info);	16
void SetTopicInfo(arTopicID topic, const arTopicInfo *info);	16
<b>Type definitions</b>	
arTypeID CreateNoteType(const char* name, Boolean userVisible);	17
Integer CountTypeInstances(arTypeID type);	17
OWN arListID GetTypeInstances(arTypeID type);	17
void GetTypeInfo(arTypeID type, IO arTypeInfo *info);	17
void SetTypeInfo(arTypeID type, const arTypeInfo *info);	17
<b>Field definitions</b>	
arFieldID CreateField( const char* name, arFieldType dataType, Boolean userVisible );	17

void GetFieldInfo(arFieldID field, IO arFieldInfo *info);	18
void SetFieldInfo(arFieldID field, const arFieldInfo *info);	18
<b>Windows</b>	
arWindowID CreateWindow(const char* name, Rect bounds);	18
arTopicID GetCurrentTopic(arWindowID window, Boolean returnRoot);	18
Boolean SetCurrentTopic( arWindowID window, arTopicID topic, Boolean useLastView );	18
Boolean SelectWindow(arWindowID window);	19
void GetWindInfo(arWindowID window, IO arWindInfo *info);	19
void SetWindInfo(arWindowID window, const arWindInfo *info);	19
<b>List Calls</b>	
Integer GetListLen(arListID list);	19
arNoteID GetListEntry(arListID list, Integer index);	19
Integer GetListEntries( arListID list, Integer index, OUT arNoteID* buffer, Integer bufSize );	20
Integer SearchList(arListID list, arNoteID target);	20
void SetListEntry(IO arListID *list, Integer index, arNoteID entry);	20
void SetListEntries( IO arListID *list, Integer index, const arNoteID* entries, Integer count );	20
void InsertListEntry(IO arListID *list, Integer index, arNoteID entry);	20
void InsertListEntries(IO arListID *list, Integer index, Integer count);	20
void DeleteListEntries(IO arListID *list, Integer index, Integer count);	20
void ResizeList(IO arListID *list, Integer newSize);	20
OWN arListID CreateList(Integer length);	21
OWN arListID DupList(arListID list);	21
void DisposeList(OWN arListID list);	21
arPathType GetPathInfo(arPathID path, OUT arNoteID *root);	21
Integer GetPathLen(arPathID path);	21
arNoteID GetPathEntry(arPathID path, Integer index);	21
Boolean PushPath(arPathID path, arNoteID newEntry);	21
void PopPath(arPathID path);	21
Boolean WalkPath( arPathID path, arDirCode direction, IO arWalkCache* cache );	21
void DisposeWalkCache(arWalkCache cache);	22
arPathID PathToTopic(arTopicID topic, Boolean atEnd);	22
Boolean EqualPaths(arPathID path1, arPathID path2);	22
arPathID DupPath(arPathID path);	22
void DisposePath(arPathID path);	22
<b>View Calls</b>	
arNoteState GetNoteState(arNoteID note, arNoteID parentNote);	23
void SetNoteState( arNoteID note, arNoteID parentNote, arNoteState newState );	23
<b>Doc Calls</b>	
arDocumentPtr GetCurrentDoc();	23
void SetCurrentDoc(arDocumentPtr newDoc);	23
Integer CountOpenDocs();	23
arDocumentPtr GetIndexedDoc(Integer index);	23
arDocumentPtr GetPrefsDoc();	23
void GetDocInfo(IO arDocInfo *info);	24
arDocumentPtr NewDoc( const FSSpec* stationeryFile, const char* title );	24
arDocumentPtr Open(const FSSpec* srcFile, Boolean readOnly);	24
void Close();	24
arFileOpResult Save();	24
arFileOpResult SaveAs(const FSSpec* targetFile);	25

arFileOpResult Revert();	25
Integer GetGlobalData(const char* name, Integer bufLen, OUT void* buffer);	25
void SetGlobalData( const char* name, const void* data, Integer dataLen, Boolean isText );	25
Boolean BringCurDocToFront();	25
void SetChangedFlag();	26
void SetupUndo(const char* operationName, Boolean setChangedFlag);	26
void OpenHiddenDoc( FSSpec* srcFile, Boolean readOnly, OUT arDocumentPtr* doc, OUT Boolean* closeWhenFinished );	26
IO Calls	
Boolean ImportTAIL(const FSSpec* srcFile, arTopicID destTopic);	26
Boolean ExportTAIL(const FSSpec* targetFile, arTopicID theTopic);	27
UI Calls	
void AddMenu(const char* menuName, Short menuCode, Integer parentCode);	27
void AddMenuItem( Short menuCode, const char* itemName, Short cmdChar, Integer commandCode, Integer itemRefcon );	27
void DeleteMenuItem( Short menuCode, Integer commandCode, Integer itemRefcon );	27
void SetMenuItem( Short menuCode, Integer commandCode, Integer itemRefcon, const char* itemName, Boolean itemEnabled, Short markChar, Short itemStyle );	27
void SetClickHook(ClickHook hook, uInteger refcon, Boolean addHook);	27
void SetMenuHook( MenuHook hook, uInteger refcon, Boolean addHook, Integer commandCode );	28
void SetKeyHook( KeyHook hook, uInteger refcon, Boolean addHook, Short charFilter, Short keyFilter, Short modFilter );	28
void SetFieldHook( FieldHook hook, uInteger refcon, Boolean addHook, arFieldID field );	29
void SetTopicHook(TopicHook hook, uInteger refcon, Boolean addHook);	29
void SetTickHook(TickHook hook, uInteger refcon, Boolean addHook);	29
void SetFileHook(FileHook hook, uInteger refcon, Boolean addHook);	29
void SetQuitHook(QuitHook hook, uInteger refcon, Boolean addHook);	29
void SetATMHook(AboutToMenuHook hook, uInteger refcon, Boolean addHook);	30
void RegisterImportFormat( const char* name, Integer formatID, OSType* fileTypes, Integer fileTypeCount, ImportHook hook, uInteger refcon );	30
void RegisterExportFormat( const char* name, Integer formatID, arEFTType type, ExportHook hook, uInteger refcon, OSType fileType, OSType fileCreator );	31
void RegisterFieldProc( const char* name, Integer id, arFPTType type, Integer fieldTypes, FieldHook proc, uInteger refcon );	31
Boolean TestFieldProc(arFieldID field, arFPTType type, Integer procID);	31
Dialog Calls	
void StopAlert(const char* message);	32
Boolean OKCancelAlert(const char* message);	32
Boolean PromptForString( const char* prompt, IO char* string, Short maxLen, Boolean numbersOnly );	32
Boolean PromptForField( const char* prompt, IO arFieldID* field, const arFieldType* allowedTypes, arChoiceDialogFlags flags );	32
Boolean PromptForType( const char* prompt, IO arTypeID* type, arChoiceDialogFlags flags );	32
void DisplayNotify(const char* message, arNotifyFlags flags);	33
void ClearNotify();	33
void SetNotifyText(const char* newMessage);	33
Boolean TickNotify(Integer amountDone, Integer totalAmount);	33
void BackgroundTick();	33

Module Manager Calls	
ModuleID FindModuleName(const char* name, uInteger minVers);	34
ModuleID FindModuleID(uInteger id, uInteger minVers);	34
const ModuleInfo* GetModInfo(ModuleID module);	34
OSErr OpenModFile(const FSSpec *file, Short fileRefnum);	34
ModuleEntryPoint GetModuleEntry(ModuleID module, uInteger entryID);	34
ProcPtr ModuleCallPrelude( ModuleEntryPoint entry, OUT ModuleParamBlock *pb );	34
void ModuleCallEpilogue( ModuleEntryPoint entry, const ModuleParamBlock *pb );	35
Memory Calls	
void* AllocMem(uInteger size, AllocMode mode);	35
void DeallocMem(void* block, AllocMode mode);	36
uInteger MemAvailable();	36
Utility Calls	
const char* GetRString(Short rsrcID, Short stringIndex);	36
void LowerString(IO char* string, Integer length);	36
void PrintToLog(const char* message);	37